Af/2152 120

Serial No.: 09/160,424
Docket No.: 1215

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

TRANSMITTAL LETTER

Inventors	:	Schneebeli, *et al.*
Serial No.	:	09/160,424
Filing Date	:	September 25, 1998
Title	:	VIRTUAL CONTENT PUBLISHING SYSTEM AND METHOD
Group/Art Unit	:	2152
Examiner	:	Willett, Stephan
Docket No.	:	1215

Mail Stop Appeal Brief - Patent
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Applicant submits herewith, in triplicate, Appellant's Appeal Brief under 37 C.F.R. § 1.192 in support of the Notice of Appeal filed on April 2, 2004, and received in the U.S. Patent and Trademark Office on April 5, 2004. Enclosed is a check in the amount of $320.00 to cover the large entity fee for filing the Appeal Brief. A duplicate copy of this Transmittal Letter is also enclosed.

Acknowledgment of receipt is respectfully requested.

Respectfully submitted,

By: Judith L. Carlson

Judith L. Carlson, Reg. No. 41,904
STINSON MORRISON HECKER LLP
1201 Walnut Street, Suite 2800
P.O. Box 419251
Kansas City, MO 64141-6251
Telephone: (816) 842-8600
Facsimile: (816) 691-3495

Certificate of Mailing Under 37 C.F.R. 1.8

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to Mail Stop Appeal Brief – Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on:

Date: June 7, 2004
Signature: Judith L. Carlson
Printed Name: JUDITH L. CARLSON

The Director is hereby authorized to charge any additional amount required, or credit any overpayment, to Deposit Account No. 19-4409.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

APPEAL BRIEF

Inventors	:	Schneebeli *et al.*
Serial No.	:	09/160,424
Filing Date	:	September 25, 1998
Title	:	VIRTUAL CONTENT PUBLISHING SYSTEM AND METHOD
Group/Art Unit	:	2152
Examiner	:	Willett, Stephan
Docket No.	:	1215

Mail Stop Appeal Brief – Patent
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 37 C.F.R. §1.192, Applicant submits this Appeal Brief in support of the Notice of Appeal filed on April 2, 2004 and received in the U.S. Patent and Trademark Office on April 5, 2004.

Certificate of Mailing Under 37 C.F.R. 1.8

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to Mail Stop Appeal Brief – Patent, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on:

Date: June 7, 2004
Signature: Judith L. Carlson
Printed Name: JUDITH L. CARLSON

The Director is hereby authorized to charge any additional amount required, or credit any overpayment, to Deposit Account No. 19-4409.

I. REAL PARTY IN INTEREST

The real party in interest in the present appeal is the assignee, Sprint Communications Company, L.P. The assignment was recorded at Reel 9885, Frame 0983 of the U.S. Patent and Trademark Office records.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-22, 24-31, 33-41, 43-46 and 48-54 are pending in the application. Claims 23, 32, 42, and 47 have been canceled. Claims 1-22, 24-31, 33-41, 43-46 and 48-54 stand finally rejected, as follows: claims 1, 14, 30, 37, 41, 46 and 51 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S Patent No. 6,026,371 to Beck *et al.* ("Beck") in view of U.S. Patent No. 6,182,111 to Inohara *et al.* ("Inohara"); claims 1, 7, 13-16, 25, 33-34, 38, 41, 46 and 52 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent 5,867,677 to Butman *et al.* ("Butman") in view of U.S. Patent No. 6,125, 388 to Reisman ("Reisman"); and claims 1-22, 24-31, 33-41, 43-46 and 48-54 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,199,082 to Ferrel *et al.* ("Ferrel") in view of U.S. Patent No. 6,134,584 to Chang *et al.* ("Chang"). The present appeal is directed to claims 1-22, 24-31, 33-41, 43-46 and 48-54, which are reproduced in Appendix A attached hereto.

IV. STATUS OF AMENDMENTS

No amendment has been filed subsequent to the final rejection dated January 28, 2004.

V. SUMMARY OF THE INVENTION

The present invention is directed to a system and method for generating, editing and/or testing staging content on a staging server, and automatically transferring the staging content from the staging server to multiple production servers residing on a computer network (*e.g.*, the Internet) in response to a publish command received on the staging server. Importantly, the transferred staging content is published on each of the production servers at substantially the same time. Each production server is then able to provide the transferred staging content to content users of the computer network (*e.g.*, users browsing the Internet) in response to requests routed to the production servers from the content users. Advantageously, all of the content users may be assured of receiving the same content (*i.e.*, the transferred staging content) regardless of which of the production servers processes their particular requests.

Preferably, access to the staging server is restricted to two access levels. Specifically, a first user associated with a first access level is allowed to control the generation, editing and/or testing of staging content on the staging server, and a second user associated with a second access level is allowed to control the transfer of the staging content from the staging server to the multiple production servers. This security feature ensures that only those individuals with the proper authorization can access the staging server to perform these tasks.

More preferably, the transferred staging content on the production servers may be replaced with the production content that was on the production servers prior to the transfer (the "prior production content") in response to a rollback command received on the staging server.

The replacement of the transferred staging content with the prior production content provides for a rollback to the previous version of the content if desired, such as if a problem is encountered with a particular production server during the transfer of content.

VI. ISSUES

The issues on appeal are as follows:

A. Whether claims 1, 14, 30, 37, 41, 46 and 51 are unpatentable under 35 U.S.C. §103(a) as being obvious over Beck in view of Inohara.

B. Whether claims 1, 7, 13-16, 25, 33-34, 38, 41, 46 and 52 are unpatentable under 35 U.S.C. §103(a) as being obvious over Butman in view of Reisman.

C. Whether claims 1-22, 24-31, 33-41, 43-46 and 48-54 are unpatentable under 35 U.S.C. §103(a) as being obvious over Ferrel in view of Chang.

VII. GROUPING OF THE CLAIMS

With respect to the rejection stated in Issue A, claims 1, 14 and 51 stand or fall together; claims 30 and 37 stand or fall together; and claims 41 and 46 stand or fall together. As discussed in Section VIII.A below, these three different groups of claims are separately patentable.

With respect to the rejection stated in Issue B, claims 1, 7, 13-16, 25 and 52 stand or fall together; claims 33, 34 and 38 stand or fall together; and claims 41 and 46 stand or fall together. As discussed in Section VIII.B below, these three different groups of claims are separately patentable.

With respect to the rejection stated in Issue C, claims 1-22, 24-29 and 51-54 stand or fall together; claims 30, 31 and 33-40 stand or fall together; and claims 41, 43-46 and 48-50 stand or

fall together. As discussed in Section VIII.C below, these three different groups of claims are separately patentable.

VIII. ARGUMENT

A. Applicant's Claims are not Obvious Over Beck in View of Inohara

The Examiner has rejected claims 1, 14, 30, 37, 41, 46 and 51 under 35 U.S.C. § 103(a) as being obvious over Beck (attached hereto as Appendix B) in view of Inohara (attached hereto as Appendix C). Beck discloses a method and system for permitting businesses and organizations to preview their customized advertisements for Web-based directory listings (such as Online Yellow Page directories) on a staging database before exportation to a production database. The method comprises creating (or revising) an advertisement using widely available HTML tools, importing the resulting HTML source and associated multi-media files into a staging database, and previewing the advertisement on the staging database as if the advertisement were running on the production database. Inohara merely discloses a method and system for managing distributed data in which a plurality of computers interconnected by a network operate to distribute, share and exchange data over the World Wide Web.

Applicant respectfully submits that a prima facie case of obviousness for rejecting these claims has not been established in that the cited references do not alone or in combination disclose or suggest the claimed invention. See In re Bell, 26 U.S.P.Q. 2d 1529, 1531 (Fed. Cir. 1993)(quoting In re Rinehart, 189 U.S.P.Q. 143,147 (C.C.P.A. 1976)) (finding that the Patent and Trademark Office's burden of establishing a prima facie case of obviousness is not met unless "the teachings from the prior art itself would appear to have suggested the claimed subject matter to a person of ordinary skill in the art.").

First, Beck and Inohara do not alone or in combination disclose or suggest the transfer of staging content from a staging server to first and second (or a plurality of) production servers for publication at substantially the same time, as required by claims 1, 14, 30, 37 and 41, 46 and 51. In the Office Action dated January 28, 2004, the Examiner argues that although Beck does not explicitly teach "transferring content at the same time to more than one production server," Inohara provides this missing limitation. As initial matter, it should be noted that the claimed limitation is not the transfer of staging content from a staging server to a plurality of production servers at substantially the same time. Rather, the claims require the transfer of staging content from a staging server to a plurality of production servers for publication at substantially the same time. Clearly, Beck does not disclose this limitation in that it merely discloses the exportation of an advertisement from a staging database to a single production database. Inohara also does not disclose this limitation. The portions of Inohara cited by the Examiner merely disclose the transfer of requests for URL content from a host server to other servers and/or the receipt at the host server of URL messages from other servers (wherein each of the URL messages is a list of specific URL contents that have been added to one of the other servers). Nowhere does Inohara disclose or suggest the transfer of any type of information (whether it be requests for URL content or URL messages) to a plurality of servers for publication at substantially the same time. Thus, claims 1, 14, 30, 37 and 41, 46 and 51 are clearly distinguishable from Beck and Inohara.

In addition, Beck and Inohara do not alone or in combination disclose or suggest limiting access to a staging server, wherein a first user associated with a first access level is allowed to control generation of staging content, and wherein a second user associated with a second access level is allowed to control the transfer of staging content from a staging server to multiple production servers, as required by claims 30 and 37. Rather, Beck discloses that an

advertisement can be edited by a variety of sources, such as a business, a publisher, or anyone providing the Web-based directory listing. Also, Beck does not disclose any details as to who controls the exportation of the advertisement to the production database or how the exportation is accomplished. Inohara also does not disclose this limitation in that it does not teach a staging server or the transfer of staging content from a staging server to multiple production servers. Thus, claims 30 and 37 are even further distinguishable from Beck and Inohara.

Furthermore, Beck and Inohara do not alone or in combination disclose or suggest replacing the staging content transferred to the plurality of production servers with prior production content for publication at substantially the same time in response to a rollback command, as required by claims 41 and 46. In fact, neither of these references teach any type of command (whether it be a "rollback command" or otherwise) that performs the function required by claims 41 and 46, namely, replacing staging content transferred to a plurality of production servers with prior production content for publication at substantially the same time. Thus, claims 41 and 46 are even further distinguishable from Beck and Inohara.

Because the Examiner has failed to meet his burden of establishing a prima facie case of obviousness, claims 1, 14, 30, 37, 41, 46 and 51 should be allowed.

B. Applicant's Claims are not Obvious Over Butman in View of Reisman

The Examiner has also rejected claims 1, 7, 13-16, 25, 33-34, 38, 41, 46 and 52 under 35 U.S.C. § 103(a) as being unpatentable over Butman (attached hereto as Appendix D) in view of Reisman (attached hereto as Appendix E). Butman discloses a system that includes a domain communications server connected over the Internet to a number of client side communications servers (which are located at the sites of member corporate clients). Information may be disseminated from any one of the client side communications servers to any or all of the other

client side communications servers through the domain communications server. This arrangement creates an intelligent extranet that links a community of member corporate clients together over the Internet via the domain communications server. As a result, each member corporate client can communicate with other member corporate clients as though the others were a part of its own internal network or intranet. Reisman merely discloses an information transport component that can be used with a variety of electronic information products (e.g., electronic magazines) to automate the mass distribution of updates (e.g., current issues) to a wide user base.

Applicant respectfully submits that a prima facie case of obviousness for rejecting these claims has not been established in that the cited references do not alone or in combination disclose or suggest the claimed invention. See In re Bell, 26 U.S.P.Q. 2d 1529, 1531 (Fed. Cir. 1993)(quoting In re Rinehart, 189 U.S.P.Q. 143,147 (C.C.P.A. 1976)) (finding that the Patent and Trademark Office's burden of establishing a prima facie case of obviousness is not met unless "the teachings from the prior art itself would appear to have suggested the claimed subject matter to a person of ordinary skill in the art.").

First, Butman and Reisman do not alone or in combination disclose or suggest the transfer of staging content from a staging server to first and second (or a plurality of) production servers for publication at substantially the same time, as required by claims 1, 7, 13-16, 25, 33-34, 38, 41, 46 and 52. In the Office Action dated January 28, 2004, the Examiner argues that Butman teaches the claimed invention "except for explicitly teaching a scheduling system," and that Reisman provides this missing limitation. This argument misses the mark. In Butman, a client side communications server is able to send information to other client side communications servers by communicating directly with an intermediate domain communications server. Neither the domain communications server nor the client side

communications servers are used as staging servers. In addition, the information sent between the client side communications servers through the domain communications server is not published at substantially the same time. The Reisman reference merely discloses an information transport component that can be used to automate the mass distribution of updates to a wide user base. It does not disclose a staging server or the transfer of information to multiple production servers for publication at substantially the same time. Thus, claims 1, 7, 13-16, 25, 33-34, 38, 41, 46 and 52 are clearly distinguishable from Butman and Reisman.

In addition, Butman and Reisman do not alone or in combination disclose or suggest limiting access to a staging server, wherein a first user associated with a first access level is allowed to control generation of staging content, and wherein a second user associated with a second access level is allowed to control the transfer of staging content from a staging server to multiple production servers, as required by claims 33, 34 and 38. Simply stated, neither Butman nor Reisman disclose this limitation because they do not teach a staging server or the transfer of staging content from a staging server to multiple production servers. Thus, claims 33, 34 and 38 are even further distinguishable from Butman and Reisman.

Furthermore, Butman and Reisman do not alone or in combination disclose or suggest replacing the staging content transferred to the plurality of production servers with prior production content for publication at substantially the same time in response to a rollback command, as required by claims 41 and 46. In fact, neither of these references teach any type of command (whether it be a "rollback command" or otherwise) that performs the function required by claims 41 and 46, namely, replacing staging content transferred to a plurality of production servers with prior production content for publication at substantially the same time. Thus, claims 41 and 46 are even further distinguishable from Butman and Reisman.

Because the Examiner has failed to meet his burden of establishing a prima facie case of obviousness, claims 1, 7, 13-16, 25, 33-34, 38, 41, 46 and 52 should be allowed.

C. Applicant's Claims are not Obvious Over Ferrel in View of Chang

The Examiner has further rejected claims 1-22, 24-31, 33-41, 43-46 and 48-54 under 35 U.S.C. § 103(a) as being obvious over Ferrel (attached hereto as Appendix F) in view of Chang (attached hereto as Appendix G). Ferrel discloses a multimedia publishing system that can be used to publish on-line newspapers, magazines and the like. In this system, two different components of a publication (namely, the layout of the publication and the content of the publication) are uploaded and stored separately on a server located at a public distribution point. The upload of the layout component of the publication to the public distribution point is performed on a limited basis (e.g., only upon initial creation of the publication) due to the fact that a publication's layout typically remains constant. However, because the content typically changes, the content component of the publication is uploaded to the public distribution point on a regular basis. In operation, when an end user initially downloads the publication, both the content and the layout components of the publication are transmitted to the end user's computer. Subsequent downloads, however, transmit only the content component of the publication to the end user's computer because the layout component has been cached on the end user's computer after the initial download. Ferrel discloses that this publication scheme allows for the download of a publication in bandwidth limited environments due to the fact that the layout component of the publication (which is typically bandwidth intensive) does not need to be transmitted to the end user after the initial download. Chang merely discloses a system and method that allows an end user to schedule the download of data such as web pages, databases or software, over a computer network such as the Internet.

Applicant respectfully submits that a prima facie case of obviousness for rejecting these claims has not been established in that the cited references do not alone or in combination disclose or suggest the claimed invention. See In re Bell, 26 U.S.P.Q. 2d 1529, 1531 (Fed. Cir. 1993)(quoting In re Rinehart, 189 U.S.P.Q. 143,147 (C.C.P.A. 1976)) (finding that the Patent and Trademark Office's burden of establishing a prima facie case of obviousness is not met unless "the teachings from the prior art itself would appear to have suggested the claimed subject matter to a person of ordinary skill in the art.").

First, Ferrel and Chang do not alone or in combination disclose or suggest the transfer of staging content from a staging server to first and second (or a plurality of) production servers for publication at substantially the same time, as required by claims 1-22, 24-31, 33-41, 43-46 and 48-54. In the Office Action dated January 28, 2004, the Examiner argues that Ferrel teaches the claimed invention "except for explicitly teaching a scheduling system," and that Chang provides this missing limitation. Yet again, this argument misses the mark. Ferrel discloses a multimedia publishing system that includes a server located at a public distribution point that stores the layout and content components of a publication. Importantly, the layout and content components of the publication are transferred to a single public distribution point (i.e., a production server), at which point the components are separately available to end users surfing the Internet. The layout and content components of the publication are not, however, transferred to a plurality of production servers, let alone for publication at substantially the same time. As to the Chang reference, it merely discloses a system that allows an end user to schedule the download of data over the Internet. It does not disclose the transfer of information to multiple production servers for publication at substantially the same time. Thus, claims 1-22, 24-31, 33-41, 43-46 and 48-54 are clearly distinguishable from Ferrel and Chang.

In addition, Ferrel and Chang do not alone or in combination disclose or suggest limiting access to a staging server, wherein a first user associated with a first access level is allowed to control generation of staging content, and wherein a second user associated with a second access level is allowed to control the transfer of staging content from a staging server to multiple production servers, as required by claims 30, 31 and 33-40. Furthermore, Ferrel and Chang do not alone or in combination disclose or suggest replacing the staging content transferred to the plurality of production servers with prior production content for publication at substantially the same time in response to a rollback command, as required by claims 41, 43-46 and 48-50. Thus, these claims are even further distinguishable from Ferrel and Chang.

Because the Examiner has failed to meet his burden of establishing a prima facie case of obviousness, claims 1-22, 24-31, 33-41, 43-46 and 48-54 should be allowed.

IX. APPENDICES

Attached hereto are the following Appendices:

Appendix A – Claims on Appeal

Appendix B – U.S. Patent No. 6,026,371 to Beck *et al.*

Appendix C – U.S. Patent No. 6,182,111 to Inohara *et al.*

Appendix D – U.S. Patent No. 5,867,667 to Butman *et al.*

Appendix E – U.S. Patent No. 6,125,388 to Reisman

Appendix F – U.S. Patent No. 6,199,082 to Ferrel *et al.*

Appendix G – U.S. Patent No. 6,134,584 to Chang *et al.*

X. SUMMARY

For the foregoing reasons, Applicant respectfully submits that claims 1-22, 24-31, 33-41, 43-46 and 48-54 are patentable over the cited references and should be allowed. Accordingly, Applicant respectfully requests that the Board reverse the Examiner's rejections and allow claims 1-22, 24-31, 33-41, 43-46 and 48-54.

Respectfully submitted,

By: Judith L. Carlson

Judith L. Carlson, Reg. No. 41,904
STINSON MORRISON HECKER LLP
1201 Walnut Street, Suite 2800
P.O. Box 419251
Kansas City, MO 64141-6251
Telephone: (816) 842-8600
Facsimile: (816) 691-3495

APPENDIX A

Claims on Appeal

1. A system for publishing network content, the system comprising:

(a) first and second production servers wherein each production server provides production content to content users of a computer network in response to requests routed to the production server from the content users;

(b) a staging server operatively connected to each of the first and second production servers, wherein staging content is generated, edited and/or tested by an administrator on the staging server and wherein the staging content is automatically transferred from the staging server to the first and second production servers for publication on the first and second production servers at substantially the same time in response to a publish command received on the staging server, wherein the transferred staging content published on each of the production servers is the same staging content; and

(c) wherein the transferred staging content replaces the production content on the production server such that the transferred staging content becomes subsequent production content accessible by the content users of the computer network, and wherein access to the staging server is limited such that the staging content is not accessible by the content users prior to the transfer to the production server.

2. The system of Claim 1 further comprising a file server for storing the staging content.

3. The system of Claim 1 further comprising a firewall operable to limit access to the staging server.

4. The system of Claim 3 wherein the staging server comprises a segmented server providing processing for a plurality of users.

5. The system of Claim 3 wherein:

a same address is associated with the first production server and the staging server; and

requests associated with the same address are routed to the staging server in response to access through the firewall.

6. The system of Claim 1 wherein the staging server is operable to generate requests for additional content from the network.

7. The system of Claim 1 wherein the staging server is operable to schedule said transfer of the staging content.

8. The system of Claim 7 wherein the staging server is operable to cancel said scheduled transfer.

9. The system of Claim 1 wherein the staging server is operable to replace the production content with prior production content, the prior production content comprising production content previously transferred to the first production server.

10. The system of Claim 1 wherein the staging server is operable to prevent alteration of the staging content on the staging server.

11. The system of Claim 1 wherein the staging server is operable to provide information selected from the group consisting of: log files, status information and combinations thereof.

12. The system of Claim 1 wherein the staging server is operable to provide user selections for at least two actions selected from the group consisting of:

generating requests for additional content from the network;

scheduling said transfer of the staging content;

canceling said scheduled transfer;

replacing the production content with prior production content and controlling saving of the production content;

preventing alteration of the staging content on the staging server, and

providing information selected from the group consisting of: log files, status information and combinations thereof.

13. The system of Claim 1 wherein the first production server is geographically remote from the second production server.

14. A method for publishing content on a computer network, the method comprising the steps of:

(a) providing a staging server wherein staging content is generated, edited and/or tested by an administrator on the staging server;

(b) limiting access to the staging server such that the staging content is not accessible by content users of the computer network;

(c) receiving a publish command on the staging server;

(d) automatically transferring the staging content from the staging server to first and second production servers for publication on the first and second production servers at substantially the same time in response to step (c), wherein the transferred staging content published on each of the production servers is the same staging content;

(e) replacing production content on the first and second production servers with the transferred staging content such that the transferred staging content becomes subsequent production content; and

(f) providing the subsequent production content to the content users of the computer network in response to requests routed to either of the first and second production servers from the content users.

15. The method of Claim 14 further comprising:

(g) storing the staging content on a file server.

16. The method of Claim 15 wherein step (g) comprises storing the staging content prior to performing step (d).

17. The method of Claim 14 further comprising:

(g) verifying a user for access to the staging server.

18. The method of Claim 17 further comprising:

(h) segmenting step (a) for a plurality of administrators.

19. The method of Claim 17:

wherein a same address is associated with the staging server and the first production server;

further comprising: (h) routing requests to the staging server in response to step (g).

20. The method of Claim 17 wherein step (g) comprises verifying access by the user as one of at least two access levels.

21. The method of Claim 20 further comprising step (g) of limiting control of step (c) to a first of the at least two access levels.

22. The method of Claim 20 further comprising step (g) of limiting control of step (a) to the administrator.

24. The method of Claim 23 wherein step (g) comprises generating requests for additional content from the computer network from the staging server.

25. The method of Claim 14 further comprising step (g) of scheduling step (d) from the staging server.

26. The method of Claim 25 wherein step (g) comprises canceling step (d).

27. The method of Claim 14 further comprising:

(g) receiving a replace content command; and

(h) replacing the production content on the first and second production servers with prior production content in response to step (g), the prior production content comprising content previously on the first and second production servers.

28. The method of Claim 14 further comprising step (g) of providing information selected from the group consisting of: log files, status information and combinations thereof in the staging server.

29. The method of Claim 14 further comprising providing user selections for at least two actions selected from the group consisting of:

testing the interaction of the staging content with the computer network from the staging server;

scheduling step (d);

canceling said scheduled transfer;

replacing the production content on the first and second production servers with prior production content, the prior production content comprising production content previously on the first and second production servers;

preventing alteration of the staging content on the staging server by a content user; and

providing information selected from the group consisting of: log files, status information and combinations thereof.

30. A method for publishing content on a computer network, the method comprising the steps of:

(a) providing a staging server on the computer network;

(b) limiting access to the staging server such that the server is not accessible by content users of the computer network, the access comprising at least first and second access levels;

(c) generating staging content on the staging server;

(d) restricting step (c) in response to a command associated with the first access level;

(e) receiving a publish command on the staging server;

(f) automatically transferring the generated staging content from the staging server to first and second production servers for publication on the first and second production servers at substantially the same time in response to step (e), wherein the transferred staging content published on each of the production servers is the same staging content; and

(g) restricting step (f) in response to a command associated with the second access level.

31. The method of Claim 30 wherein step (c) comprises editing the staging content.

33. The method of Claim 30 further comprising:

(h) replacing production content on the first and second production servers with the transferred staging content of step (f); and

(i) reversing step (h).

34. The method of Claim 30 wherein step (f) comprises replacing the transferred staging content of step (f) on the first and second production servers with the production content.

35. The method of Claim 30 wherein:

the staging server includes segmented software; and

step (c) comprises generating staging content for each of a plurality of administrators, each administrator associated with a segment of the segmented software.

36. The method of Claim 30 further comprising providing user selections for at least two actions selected from the group consisting of:

testing the interaction of the staging content with the computer network from the staging server;

scheduling a transfer of the staging content to the first production server and the second production server;

canceling said scheduled transfer;

transferring the staging content to the first production server and the second production server in response to a publish command;

replacing production content on the first production server with prior production content, the prior production content comprising content previously on the first production server and the second production server;

preventing alteration of the staging content on the staging server by a user associated with a second of the at least two access levels; and

providing information selected from the group consisting of: log files, status information and combinations thereof.

37. A system for publishing content on a computer network, the system comprising:

a staging server and associated software comprising a staging area on the computer network, the staging area operable to allow generation, editing and/or testing by an administrator of staging content and transfer of the staging content from the staging area to a plurality of production areas for publication on the production areas at substantially the same time, wherein the transferred staging content published on each of the production areas is the same staging content;

a firewall operable to limit access to the staging area to at least two access levels such that the staging area is not accessible by content users of the computer network, the firewall operatively connected to the staging server; and

wherein a first user associated with a first of the at least two access levels is allowed to control generation, editing and/or testing of the staging content, and wherein a second user associated with a second of the at least two access levels is allowed to control transfer of the staging content from the staging area to the production areas.

38. The system of Claim 37 further comprising a production server associated with production content; and

wherein the production content is replaced with the staging content associated with the staging area and the replacement is reversed at a later time.

39. The system of Claim 37 wherein the software comprises segmented software; and the each segment of the segmented software is associated with one of a plurality of user groups.

40. The system of Claim 37 further comprising a user interface associated with selections for at least two actions selected from the group consisting of:

testing the interaction of the staging content with the computer network from the staging area;

scheduling a transfer of the staging content to a first production server and a second production server;

canceling said scheduled transfer;

transferring the content to the first production server and the second production server in response to a publish command;

replacing production content on the first production server and the second production server with prior production content, the prior production content comprising content previously on the first production server and the second production server;

providing information selected from the group consisting of: log files, status information and combinations thereof.

41. A method for publishing content on a computer network, the method comprising the steps of:

(a) providing a staging server and a plurality of production servers on the computer network, the staging server associated with staging content and each of the production servers associated with production content, wherein the staging content is not accessible on the staging server by content users of the computer network;

(b) replacing the production content on each of the production servers with the staging content for publication on the production servers at substantially the same time in response to a publish command associated with the staging server, wherein the staging content published on each of the production servers is the same staging content, whereby the staging content becomes accessible on the production servers by the content users of the computer network; and

(c) replacing the staging content on each of the production servers with the production content for publication on the production servers at substantially the same time in response to a rollback command associated with the staging server, wherein the production content published on each of the production servers is the same production content, whereby the production content is accessible on the production servers by the content users of the computer network.

43. The method of Claim 41 further comprising:

(d) limiting access to the staging server to at least two access levels;

(e) generating the staging content on the staging server; and

(f) restricting step (e) in response to a command associated with one of the at least two access levels.

44. The method of Claim 41 wherein:

the staging server includes segmented software; and

further comprising (d) generating staging content for each of a plurality of administrators, each administrator associated with a segment of the segmented software.

45. The method of Claim 41 further comprising providing user selections associated with at least two actions selected from the group consisting of:

testing an interaction of the staging content with the computer network from the staging server;

scheduling a transfer of the staging content to a first production server;

canceling said scheduled transfer;

transferring the staging content to the first production server and the second production server in response to a publish command;

preventing alteration of the staging content by a user associated with a first access level; and

providing information selected from the group consisting of: log files, status information and combinations thereof.

46. A system for publishing content on a computer network, the system comprising:

a staging server associated with the computer network and with staging content, wherein access to the staging server is limited such that the staging content is not accessible by content users of the computer network;

a plurality of production servers wherein each production server is associated with the computer network and with production content that is accessible by the content users of the computer network;

a staging server user interface that allows a user to select a publish command associated with replacement of the production content on each of the production servers with the staging content for publication on each of the production servers at substantially the same time, wherein the staging content published on each of the production servers is the same staging content; and

wherein the staging server user interface also allows the user to select a rollback command associated with replacement of the staging content on each of the production servers with the production content for publication at substantially the same time.

48. The system of Claim 46 further comprising a firewall for limiting access to the staging server to at least two access levels; and

wherein the staging server is operable to generate the staging content in response to input associated with the staging server user interface and is operable to restrict the generation in response to a command associated with one of the at least two access levels.

49. The system of Claim 46 wherein:

the staging server includes segmented software; and

the staging server is operable to generate the staging content for each of a plurality of users, each user associated with a segment of the segmented software.

50. The system of Claim 46 wherein the staging server user interface is associated with user selections for at least two actions selected from the group consisting of:

testing an interaction of the staging content with the computer network from the staging server;

scheduling a replacement of the production content with the staging content;

canceling said scheduled replacement;

replacing the production content on the plurality of production servers with the staging content in response to the publish command;

preventing alteration of the staging content by a user associated with a first access level; and

providing information selected from the group consisting of: log files, status information and combinations thereof.

51. A method for publishing content on a computer network, the method comprising the steps of:

(a) generating, editing and/or testing staging content by an administrator on a staging server, wherein access to the staging server is limited such that the staging content is not accessible on the staging server by content users of the computer network;

(b) replicating the staging content to at least first and second temporary directories;

(c) transferring the staging content from the staging server to first and second production servers associated with the first and second temporary directories, respectively, for publication on the first and second production servers at substantially the same time, wherein the transferred staging content published on the production servers is the same staging content; and

(d) providing the transferred staging content to the content users of the computer network in response to requests routed to either of the first and second production servers from the content users.

52. The method of Claim 51 further comprising step (e) of commanding publication, wherein steps (b) and (c) are responsive to step (e).

53. The method of Claim 51 further comprising step (e) of verifying step (b).

54. The method of Claim 53 wherein step (c) is responsive to step (e).

APPENDIX B

U.S. Patent No. 6,026,371 to Beck *et al.*

US006026371A

United States Patent [19]

Beck et al.

[11] Patent Number: 6,026,371

[45] Date of Patent: Feb. 15, 2000

[54] **METHOD AND APPARATUS FOR ALLOWING ONLINE DIRECTORY PRODUCERS TO PREVIEW ADVERTISEMENT IN ONLINE DIRECTORY LISTINGS**

[75] Inventors: **Teresa Marie Beck**, Sunrise, Fla.; **Gerald Eugene Haegele**, Raleigh, N.C.; **Wendi Lynn Nusbickel**, Delray Beach, Fla.

[73] Assignee: **International Business Machines Corp.**, Armonk, N.Y.

[21] Appl. No.: **08/978,240**

[22] Filed: **Nov. 25, 1997**

[51] **Int. Cl.**[7] **G06F 17/60**

[52] **U.S. Cl.** **705/14**; 707/10; 709/203; 709/219

[58] **Field of Search** 705/1, 10, 14; 395/200.31, 200.33, 200.35, 200.47, 200.48, 200.49; 707/201, 204, 501, 513, 10; 709/201, 203, 205, 217, 218, 219

[56] **References Cited**

U.S. PATENT DOCUMENTS

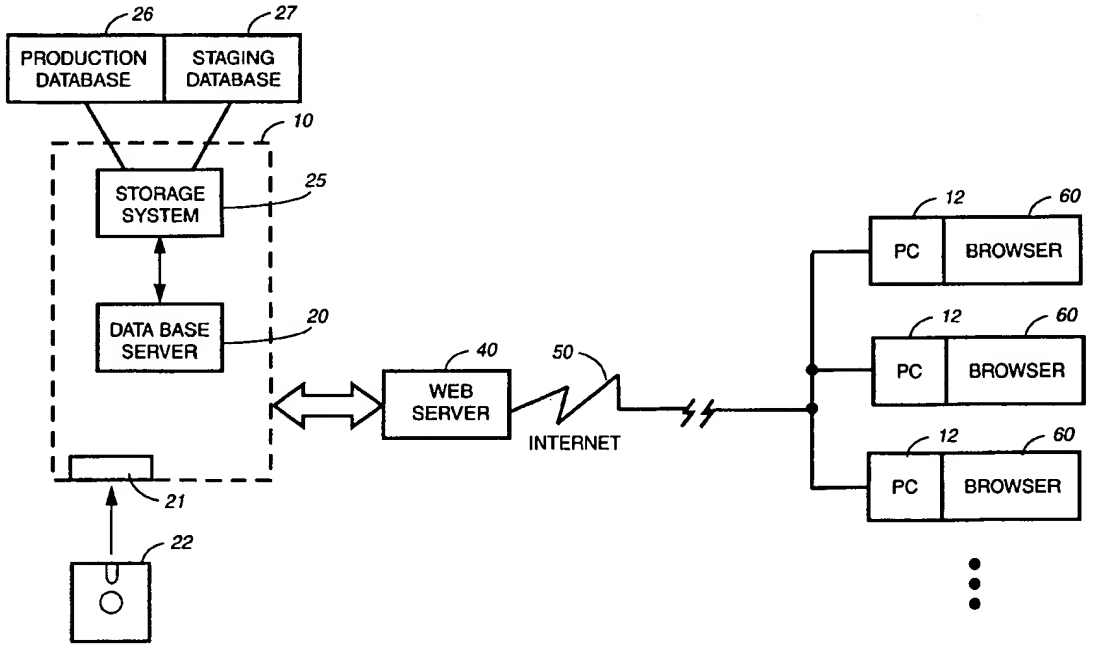
5,793,966	8/1998	Amstein et al.	395/200.33
5,829,001	10/1998	Li et al.	707/10
5,832,487	11/1998	Olds et al.	707/10
5,845,299	12/1998	Arora et al.	707/513
5,870,552	2/1999	Dozier et al.	395/200.49

Primary Examiner—Allen R. MacDonald
Assistant Examiner—Hlani M. Kazimi
Attorney, Agent, or Firm—Gunster, Yoakley, Valdes-Fauli & Stewart, P.A.; Jon A. Gibbons

[57] **ABSTRACT**

A method and system for permitting businesses and organizations to preview their customized Web-based advertisement in an Online directory listing. The method consists of: linking at least one database server to at least one Web server. The database server has a production database and a staging database wherein the staging database is a replica copy of the production database. Creating (or revising) custom multimedia advertisement material using a variety of widely available HTML (Hyper Text Markup Language) tools; importing the resulting HTML source and associated multimedia files into the staging database stored in the database server; and previewing the custom Web pages on the staging database as if they were running on the production database.

19 Claims, 5 Drawing Sheets



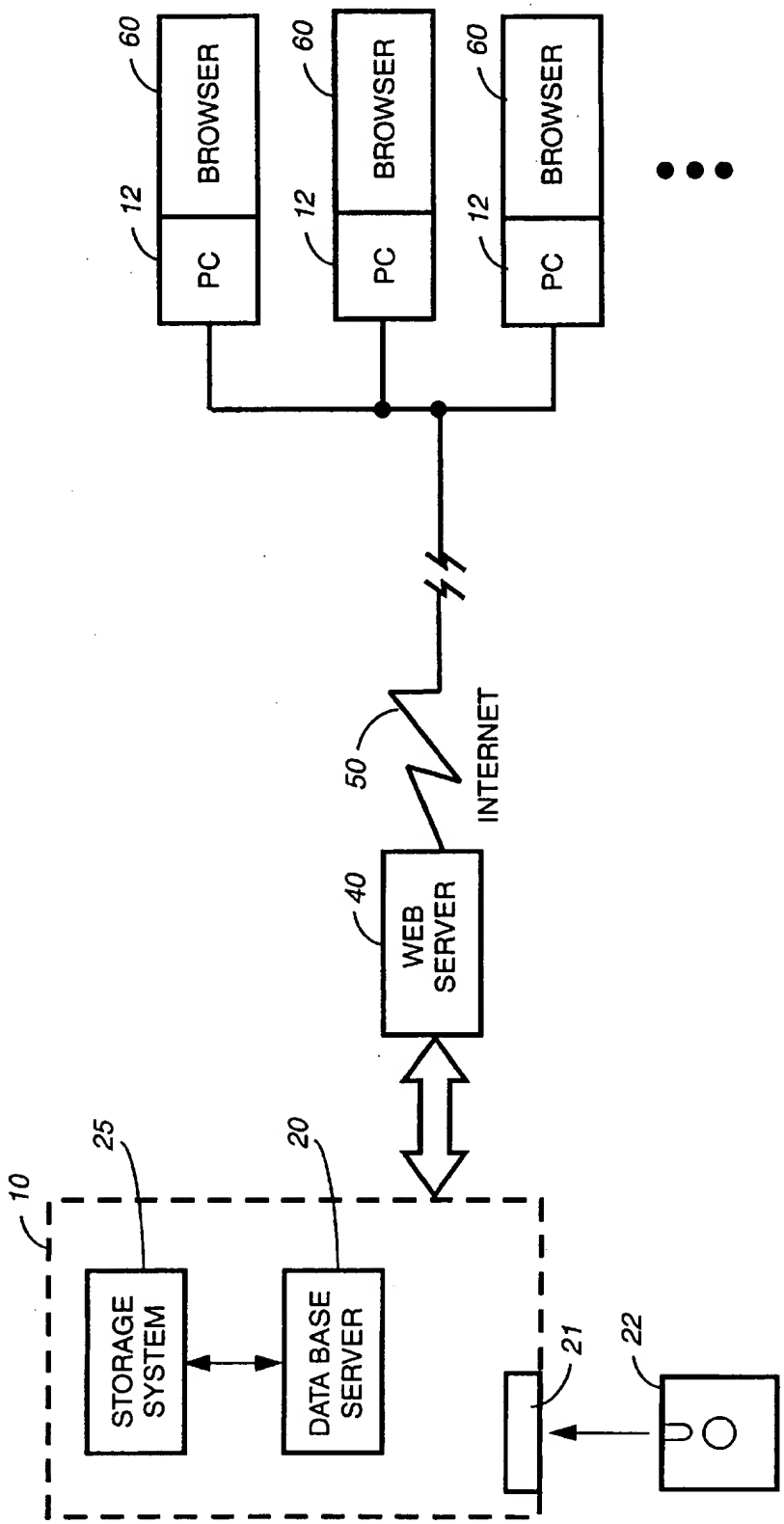

60
BROWSER
12
PC
60
BROWSER
12
PC
60
BROWSER
12
PC
50
INTERNET
40
WEB SERVER
25
20
10
STORAGE SYSTEM
DATA BASE SERVER
21
22

FIG. 1

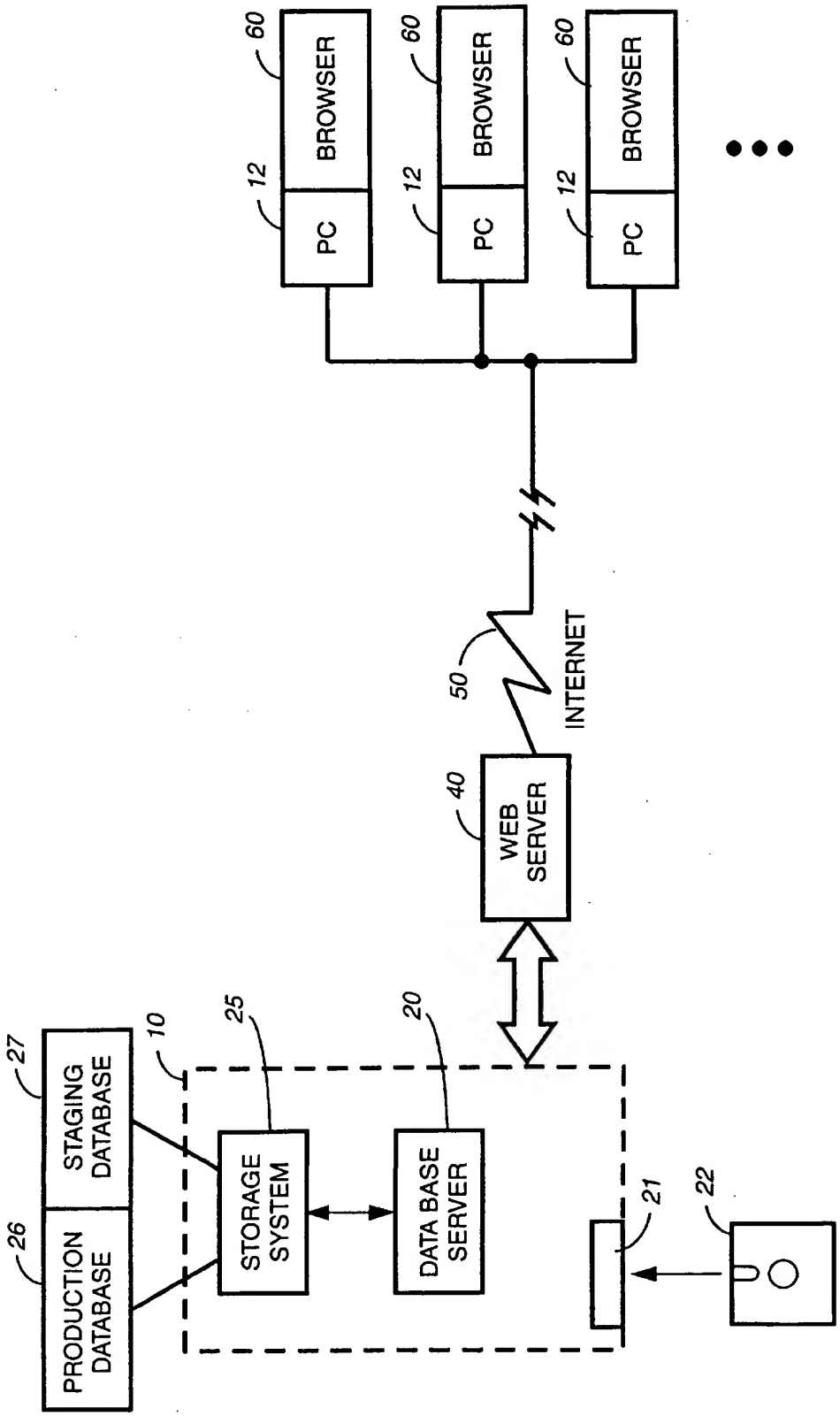

60
BROWSER
12
PC
60
BROWSER
12
PC
60
BROWSER
12
PC
50
INTERNET
40
WEB SERVER
10
25
20
27
STAGING DATABASE
26
PRODUCTION DATABASE
STORAGE SYSTEM
DATA BASE SERVER
21
22

FIG. 2

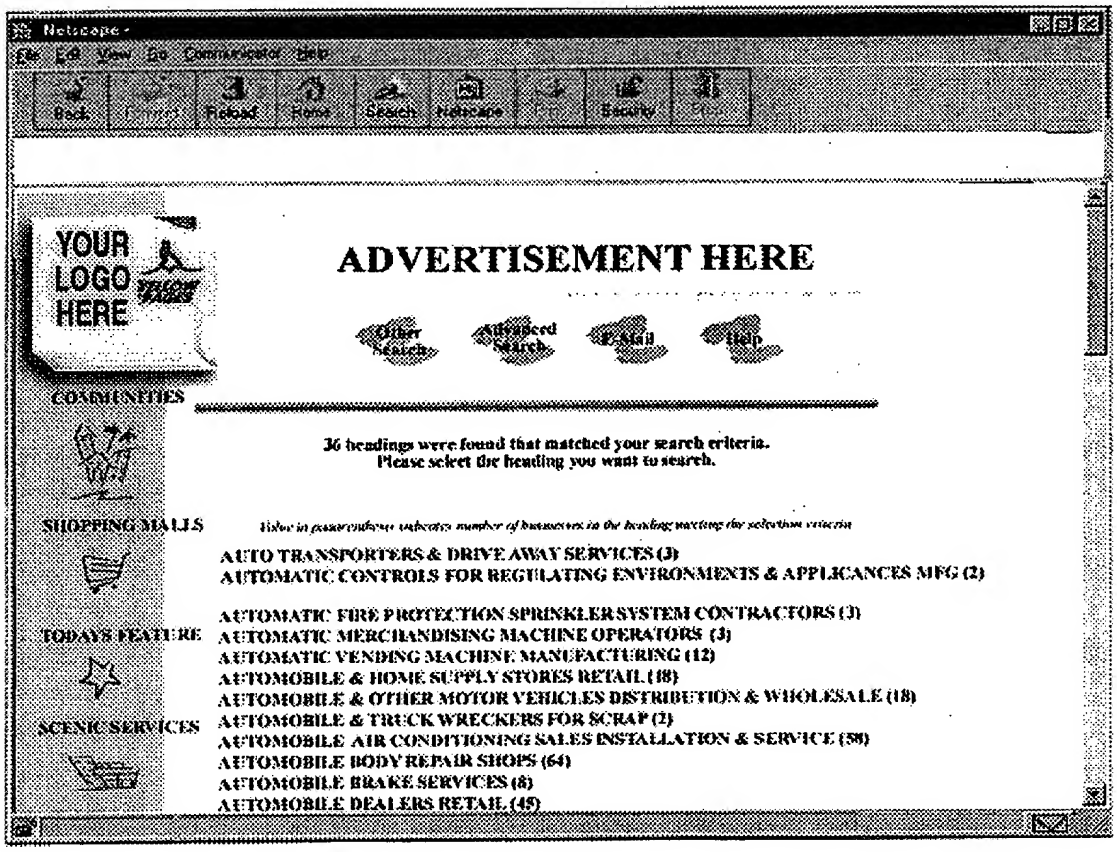

YOUR LOGO HERE
ADVERTISEMENT HERE
Other search
Advanced search
Help
Help
COMMUNITIES
SHOPPING MALLS
TODAYS FEATURE
SCENIC SERVICES
36 headings were found that matched your search criteria.
Please select the heading you want to search.
AUTO TRANSPORTERS & DRIVE AWAY SERVICES (3)
AUTOMATIC CONTROLS FOR REGULATING ENVIRONMENTS & APPLICANCES MFG (2)
AUTOMATIC FIRE PROTECTION SPRINKLER SYSTEM CONTRACTORS (3)
AUTOMATIC MERCHANDISING MACHINE OPERATORS (3)
AUTOMATIC VENDING MACHINE MANUFACTURING (12)
AUTOMOBILE & HOME SUPPLY STORES RETAIL (18)
AUTOMOBILE & OTHER MOTOR VEHICLES DISTRIBUTION & WHOLESALE (18)
AUTOMOBILE & TRUCK WRECKERS FOR SCRAP (2)
AUTOMOBILE AIR CONDITIONING SALES INSTALLATION & SERVICE (58)
AUTOMOBILE BODY REPAIR SHOPS (64)
AUTOMOBILE BRAKE SERVICES (8)
AUTOMOBILE DEALERS RETAIL (45)

FIG. 3


Netscape - [DIRECTORY OF BOCA RATON - AUTOMOBILE DEALERS RETAIL]
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Location: http://bestger.bocaraton.iom.com:80/eyp-serv?OREQ=LST&QSD-875297344159&QLOCSEL=1,FL,123000
YOUR LOGO HERE
YELLOW PAGES
Even rich people need worries.
Back Home
E-mail
Computer Tips
Related Headings
Other Sources
Advanced Search
E-mail
Help
Listing 1 - 25 of 45
More Listings
561-9989330
A NEWCAR VOLKWAGEN
Golf* Cabrio* GTI* Jetta* Jetta GLX
645 PARK OF COMMERCE WAY
BOCA RATON, FL 33487-8204
SALES and LEASING
954-4276900
ALLEN FORD
Have you driven a Ford lately?
I-95 AND LINTON BLVD.
DEERFIELD BEACH, FL 33441
Fax us at: 427-6901!
6900 rd 427-6900
Beach Linton Blvd.
ANACONDA AUTOMOTIVE DEALER
BOCA RATON, FL 33487
56102416111

FIG. 4

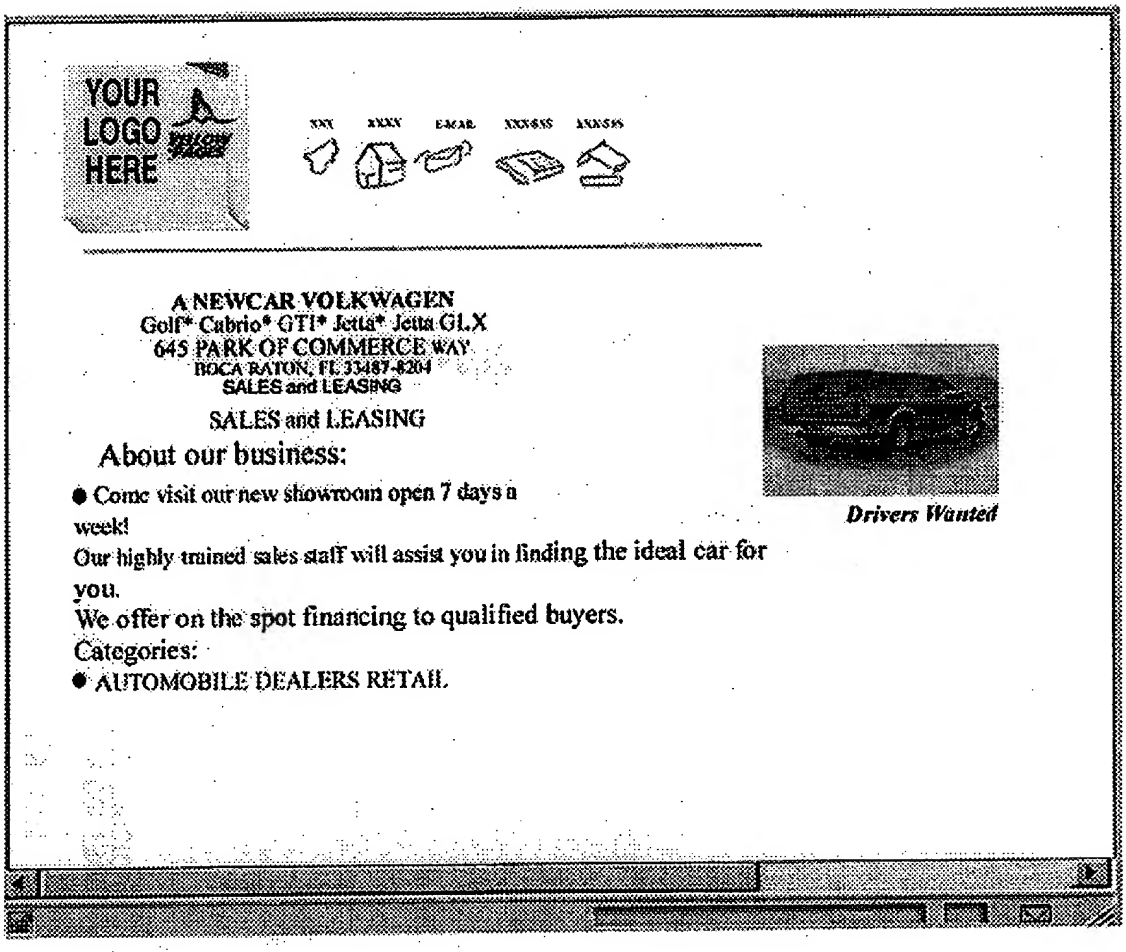

YOUR LOGO HERE
A NEWCAR VOLKWAGEN
Golf* Cabrio* GTI* Jetta* Jetta GLX
645 PARK OF COMMERCE WAY
BOCA RATON, FL 33487-8204
SALES and LEASING
SALES and LEASING
About our business:
● Come visit our new showroom open 7 days a week!
Our highly trained sales staff will assist you in finding the ideal car for you.
We offer on the spot financing to qualified buyers.
Categories:
● AUTOMOBILE DEALERS RETAIL
Drivers Wanted

FIG. 5

METHOD AND APPARATUS FOR ALLOWING ONLINE DIRECTORY PRODUCERS TO PREVIEW ADVERTISEMENT IN ONLINE DIRECTORY LISTINGS

FIELD OF THE INVENTION

The present invention relates to the field of computer network communications. More particularly, the present invention relates to Online or Web-based directory listings, such as Online Yellow Page directories, in which a method and system are disclosed that permit advertisers to preview their customized advertisements in Online directory.

BACKGROUND OF THE INVENTION

Traditional paper telephone directory listings such as the White Pages, Yellow Pages and industry-specific directory listings are well known. Online or Web-based directory listings are the Online analogues to their familiar, traditional paper counterparts. With the advent of the Internet and the Web (World-Wide-Web), many owners and publishers of these directories have begun to offer their services Online. These Online directory services are expanding beyond simply providing name, address and telephone information and have begun to offer E-mail directory listings, Web page address listings, fax directory listings, consumer tips directory listings, emergency provider directory listings and much more.

As in the traditional paper counterparts, the publishers of these Online directory listings sell advertising space to businesses and organizations to cover the expense of compiling these directories. One of the advantages to advertisers in the Web medium over the paper medium is the use of multimedia in advertising. Multimedia technologies such as text, graphics, audio and full motion video on the Web are becoming common.

Nevertheless, one of the challenges advertisers face in providing advertisements on the Web is keeping information current. The Web advertisement, even more than its paper counterpart, is susceptible to becoming quickly stale, unappealing and out-of-date. Web advertising that is months, weeks and sometimes even days old is much less effective than advertising that is frequently revised and updated. In addition, in many industries, the product cycle times have decreased, which in turn has accelerated the requirement to keep advertisements up to date.

Another challenge many businesses face with Online directory listings is the ability to preview their own custom advertising content in the manner it will be viewed by customers over the Web. Currently text only information can be scanned for unwanted or undesirable words prior to insertion into the Online directory database. However, multimedia data, such as audio, video and still images, however, cannot be inspected with the text only previewing techniques.

Still another challenge is for the providers or the producers of Online directory listing to preview advertising content in a manner it will be view by customers over the Web.

SUMMARY OF THE INVENTION

Briefly in accordance with the present invention, a method and system for permitting businesses and organizations to preview their customized Web-based advertisement in an Online directory listing. The method consists of: linking at least one database server to at least one Web server. The database server has a production database and a staging database wherein said staging database is a replica copy of said production database. Creating (or revising) custom multimedia advertisement material using a variety of widely available HTML (Hyper Text Markup Language) tools; importing the resulting HTML source and associated multimedia files into the staging database stored in the database server; and previewing the custom Web pages on the staging database as if they were running on the production database.

BRIEF DESCRIPTION OF THE DRAWING(S)

FIG. **1** is a functional block diagram of a typical prior art data processing system for hosting Web pages.

FIG. **2** is a functional block diagram of a data processing system for hosting Web-based Online directory listing services.

FIG. **3** is an illustration of a Web browser displaying an example Online Yellow Pages directory listing for the top-level search results.

FIG. **4** is an illustration of a Web browser displaying an example Online Yellow Pages directory listing for the next level search results.

FIG. **5** is an illustration of a Web browser displaying a representative Web-based advertisement for a business listing in an example Online Yellow Pages directory listing.

DESCRIPTION OF A PREFERRED EMBODIMENT

FIG. **1** depicts a functional block diagram of a typical data processing system for hosting Web pages **5**. Web server **40** is connected to the Internet **50**. A plurality of end-user data processing systems **12** with Web browser **60** are connected to Internet **50**. Web browser **60** are client software programs based upon HTTP (Hyper-Text-Transfer-Protocol) compatible product such as Netscape Navigator, JAVA Browser or Microsoft Internet Explorer.

Referring now to FIG. **2**, there is shown an Online directory listing system **10** in accordance with the invention. Online directory listing system **10** expands the information processing system for hosting Web pages **5** of FIG. **1** through the addition of a database server **20**. Typically publishers of directory listings such as regional telephone companies provide the name, address, and telephone directory listing information for database server **20**. It is important to point out that the precise operating systems and hardware configurations of database server **20**, Web server **40**, and Web-browser **60** are not limited to any specific hardware or software configuration. These systems can be implemented on a wide variety of hardware and software platforms.

The Online directory listings system **10** is connected to Web browser **60**, which provides an end-user the ability to enter a desired search criteria. Web browser **60** through Internet **50**, communicates with the Web server **40** to query the database server **20**. The results of the query from database server **20** are then sent back, through the Web server **40** and over Internet **50**, to the end-user browser **60**. FIG. **3** illustrates a typical Yellow Pages directory listing service interface rendered on Web browser **60**. Here the results for an example keyword query of "automobile" are depicted. Typically, unless the end-user requested a more narrow or restrictive search, the end-user will select from among the listing headings displayed. Suppose in this example the user chooses "automobile dealerships." FIG. **4**

depicts the results of this user preference. This listing in FIG. 4 contains the same basic information of the name, address and telephone number listed in the traditional paper Yellow Page counterpart. At this point all the traditional Online directory listing information has been provided to the end user.

The method for permitting business and organizations to customize their advertisement in Online directory listings consists of: creating (or revising) multimedia advertisement material using any of a widely available HTML (Hyper Text Markup Language) tools; importing the resulting HTML source and associated multimedia files into production database **26** of database server **20**; and Associating through the database server **20** the HTML source files and multimedia files to the targeted business or organization.

From the above disclosure it should be understood that the advertisement for a business or organization has now been customized in the Online directory listing. Returning to the previous "automobile" keyword example, in FIG. **5** an end-user can now proceed beyond just the traditional Online directory listings and now request to view more specific information for a particular business listings. An example of a business Web-based multimedia advertisement for an "automobile dealership" is shown. The information and content of this advertisement being created Online by the business itself and is automatically associated with its appropriate Online directory listing.

Facilitating the preview of a customized multimedia advertisement is accomplished by mirroring production database **26** with a staging database in FIG. **2**. Here the information in the staging database **27** is a replica copy of the information of in the production database **26**. A business or organization wishing to preview their customized multimedia advertisement material can select to have this material imported to the staging database **27**. The advertisement of FIG. **5** can be reviewed in the staging database **27**. An advertiser may wish to change text, displayed in FIG. **5** "A NEW CAR VOLKSWAGEN" to a different font, size or color.

Portions of a multimedia advertisement content of FIG. **5** may be modified, revised, deleted or edited by the business, the publisher or anyone else providing the Online listings. Portions may include any and all of the multimedia advertisement. The advertisement content is then riposted in staging database **27**.

Once imported into the replica staging database the advertiser is enabled to preview their material for any errors, mistakes, or other undesirable advertisement attributes. Upon completing their review, these web pages are exported over to the production database **26**. While the invention has been illustrated and described in the preferred embodiments, many modifications and changes therein may be affected by those skilled in the art. It is to be understood that the invention is not limited to the precise construction herein disclosed. Accordingly, the right is reserved to all changes and modifications coming within the true spirit and scope of the invention.

We claim:

1. An information processing system comprising:

at least one database server linked to at least one Web server, said Web server comprising a plurality of Web browser clients connected thereto, said database server comprising a production database and a staging database;

means for creating Web pages having multimedia data files and links to other Online data on at least one of said Web browser clients;

means for importing said Web pages into said staging database of said database server;

means for previewing said Web pages on said staging database on at least one of said Web browser clients;, as if said Web pages were running on said production database of said database server; and

means for updating said production database of said database server with said Web pages that have been imported into said staging database of said database server.

2. The information processing system of claim **1** wherein said means for previewing said Web pages on said staging database further comprises:

means for deleting each of said Web pages.

3. The information processing system of claim **1** wherein said means for previewing said Web pages on said staging database further comprises:

means for updating said production data base with said Web pages.

4. The information processing system of claim **1** further comprising:

said means for creating said Web pages on a separate information processing system and not connected to said Web server.

5. The information processing system of claim **1** wherein said means for importing includes importing of said Web pages submitted from said Web browser clients.

6. A method for previewing Web-based advertisements in an Online directory listing, comprising the steps of:

linking at least one database server to at least one Web server, said database server comprising a production database and a staging database;

creating Web pages having multimedia data files and links to other Online data on at least one of said Web browser clients;

importing Web pages into said staging database of said database server;

previewing said Web pages on said staging database on at least one of said Web browser clients as if said Web pages were running on said production database of said database server; and

updating said production database of said database server with said Web pages that have been imported into said staging database of said database server.

7. The method for previewing custom Web-based advertisements in Online directory listings of claim **6** further comprising the steps of:

selectively deleting portions of said custom multimedia Web pages.

8. The method for previewing custom Web-based advertisements in Online directory listings of claim **6** further comprising the steps of:

updating said production data base with said custom multimedia Web pages.

9. The method for previewing custom Web-based advertisements in Online directory listings of claim **6** further comprising the steps of:

creating on a separate information processing system custom multimedia Web pages having associated multimedia data files and associated links to other Online data.

10. The method for previewing custom Web-based advertisements in Online directory listings of claim **6** further comprising the steps of:

importing custom Web pages submitted from said Web browser clients having associated multimedia data files

and associated links to other Online data into said database server.

11. The method for previewing custom Web-based advertisements in Online directory listings of claim 6 further comprising the steps of:

importing pointers pointing to said custom Web pages having associated multimedia data files and associated links to other Online data into said database server such that said Web pages are hosted on another information processing system.

12. A computer-readable storage medium containing instructions for previewing custom Web-based advertisements in an Online directory listing comprising the instructions of:

linking at least one database server to at least one Web server, said database server comprising a production database and a staging database;

creating custom multimedia Web pages having multimedia data files and links to other Online data on at least one of said Web browser clients;

importing said Web pages into said staging database of said database server;

previewing said Web pages on said staging database on at least one of said Web browser clients as if said Web pages were running on said production database of said database server; and

updating said production database of said database server with said Web pages that have been imported into said staging database of said database server.

13. The computer-readable storage medium of claim 12 further comprises the instruction for:

selectively deleting portions of said Web pages.

14. The computer-readable storage medium of claim 12 further comprising the instruction for:

updating said production data base with said Web pages.

15. The computer-readable storage medium of claim 12 further comprising the instruction for:

creating on a separate information processing system Web pages having multimedia data files and links to other Online data.

16. The computer-readable storage medium of claim 12 further comprises the instruction for:

importing Web pages submitted from said Web browser clients having multimedia data files and links to other Online data into said database server.

17. The computer-readable storage medium of claim 12 further comprises the instruction for:

importing pointers pointing to said Web pages having multimedia data files and links to other Online data into said database server such that said Web pages are hosted on another information processing system.

18. An information processing system comprising:

at least one Web server;

a plurality of Web browser clients connected to said Web server;

at least one database server linked to said Web server;

a first interface in said database server for coupling to a production database;

a second interface in said database server for coupling to a staging database;

means for importing from one or more Web browser clients into said staging database custom multimedia Web pages having multimedia data files and links to other Online data;

means for updating said production database of said database server with said Web pages that have been imported into said staging database of said database server; and

means for receiving an authorization on said database server from said Web browser clients to update said production database on said database server from said staging database on said database server during previewing of said Web pages on said staging database of said database server, wherein said previewing is presented as if said Web pages were running on said production database of said database server.

19. A method for allowing online directory producers to preview their custom multimedia Web pages in an online directory system comprising the steps of:

linking at least one database server at least one Web server wherein said Web server is coupled to one or more clients;

coupling a first interface to a production database;

coupling a second interface to a staging database;

creating custom multimedia Web pages having multimedia data files and links to other Online data on said clients;

importing said Web pages from said clients into said staging database of said database server;

presenting said Web pages on said clients, wherein said staging database presents said Web pages as if said Web pages were running on said production database of said database server;

updating said production database of said database server with said Web pages that have been imported into said staging database of said database server; and

receiving an authorization from said client after previewing said Web pages to update said production database of said database server by moving said Web pages from said staging database of said database server over to said production database of said database server.

* * * * *

APPENDIX C

U.S. Patent No. 6,182,111 to Inohara *et al.*

US006182111B1

(12) **United States Patent**
Inohara et al.

(10) **Patent No.: US 6,182,111 B1**
(45) **Date of Patent: Jan. 30, 2001**

(54) **METHOD AND SYSTEM FOR MANAGING DISTRIBUTED DATA**

(75) Inventors: **Shigekazu Inohara**, Kokubunji; **Toyohiko Kagimasa**, Yokohama; **Yoshimasa Masuoka**, Kodaira; **Fumio Noda**, Kodaira; **Jinghua Min**, Kodaira, all of (JP)

(73) Assignee: **Hitachi, Ltd.**, Tokyo (JP)

(*) Notice: Under 35 U.S.C. 154(b), the term of this patent shall be extended for 0 days.

(21) Appl. No.: **09/079,151**

(22) Filed: **May 15, 1998**

(30) **Foreign Application Priority Data**

May 15, 1997 (JP) 9-125247

(51) **Int. Cl.**[7] **G06F 15/16**

(52) **U.S. Cl.** **709/201**; 709/202; 709/203; 709/217; 709/219; 709/223; 707/10

(58) **Field of Search** 709/225, 226, 709/233, 235, 238, 239, 241, 244, 200–204, 216–220, 223–224, 227–228; 707/1, 9–10, 100, 104

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,548,724	*	8/1996	Akizawa et al.	709/203
5,712,981	*	1/1998	McKee et al.	709/241
5,764,908	*	6/1998	Shoji et al.	709/217
5,774,660	*	6/1998	Brendel et al.	709/201
5,864,670	*	6/1999	Hayashi et al.	709/204
5,913,041	*	6/1998	Ramanathan et al.	709/233
5,915,095	*	6/1998	Miskowie	709/223
6,085,239	*	7/2000	Kubo et al.	709/223

OTHER PUBLICATIONS

Sape Mullender etc. "Distributed Systems (1st ed.)", pp. 13–15, ACM press, 1989.

A. Chankhunthod etc, "A Hierarchical Internet Object Cache", 1996 USENIX Technical Conference, pp. 153–163, 1996.

B. Kantor etc. "Network News Transfer Protocol: A Proposed Standard for the Stream–Based Transmission of News" Network Working Group RFC–977, Feb. 1986.

P. Mockapetris "Domain Names–Implementation and Specification" Network Working Group RFC–1035, Nov. 1987.

Michael Dahlin etc. "Cooperative Caching: Using Remote Client Memory to Improve File System Performance", First USENIX Symposium on Operating Systems Design and Implementation, 1994.

M. J. Feeley etc."Implementing Global Memory Management in an Workstation Cluster", ACM 15th Symposium on Operating Systems Principles,pp. 201–212, 1995.

P.Sarkar etc. "Efficient Cooperative Caching Using Hints", Second USENIX Symposium on Operating Systems Design and Implementation,pp. 35–46, 1996.

Jeffrey Kuskin etc."The Stanford FLASH Multiprocessor", Proceedings of the 21st Annual International Symposium on Computer Architecture,pp. 302–314,ACM, 1994.

Henk L. Muller etc."The Data Diffusion Machine with a Scalable Point–to–Point Network", Technical Report CSTR–93–17, Department of Computer Science, University of Bristol, Oct. 1993.

* cited by examiner

Primary Examiner—Robert B. Harrell
Assistant Examiner—Bharat Barot
(74) *Attorney, Agent, or Firm*—Antonelli, Terry, Stout & Kraus, LLP

(57) **ABSTRACT**

Irregular and unstable natures of the Internet to be caused by an increase in Internet accessing users are alleviated and services of an information system more comfortable to users are provided. To this end, each servers among a plurality of servers cooperating to provide services stores the past communications line state (throughput and latency), and in accordance with the stored communications lines state, cache reference and prefetch are preformed between optional servers.

29 Claims, 8 Drawing Sheets

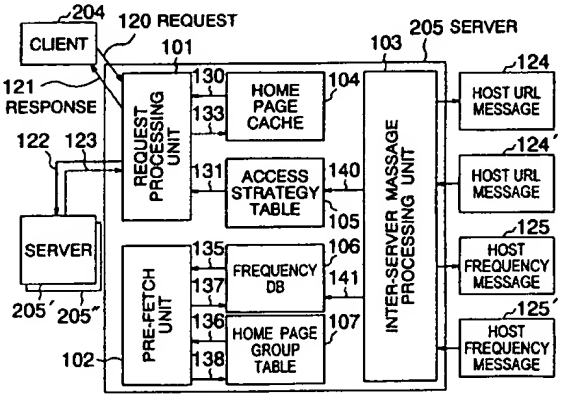


FIG. 1

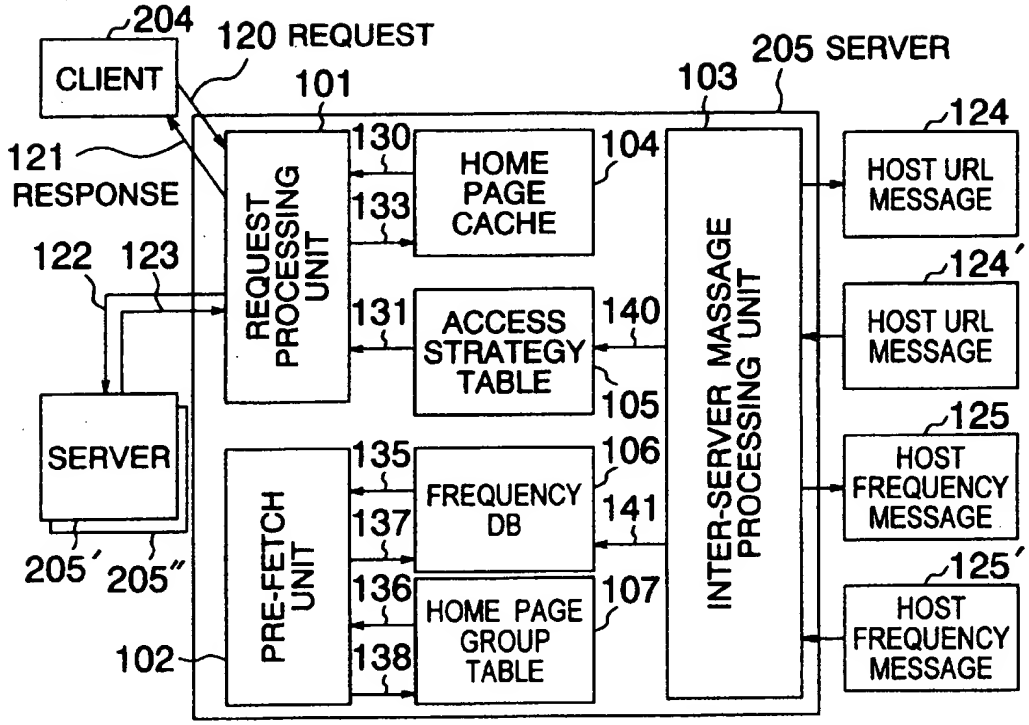

204
CLIENT
120 REQUEST
101
205 SERVER
103
121
RESPONSE
130
133
HOME PAGE CACHE
104
124
HOST URL MESSAGE
122
123
REQUEST PROCESSING UNIT
131
ACCESS STRATEGY TABLE
140
105
124'
HOST URL MESSAGE
SERVER
135
137
FREQUENCY DB
106
141
125
HOST FREQUENCY MESSAGE
205'
205"
136
138
HOME PAGE GROUP TABLE
107
125'
HOST FREQUENCY MESSAGE
102
PRE-FETCH UNIT
INTER-SERVER MASSAGE PROCESSING UNIT

FIG. 2

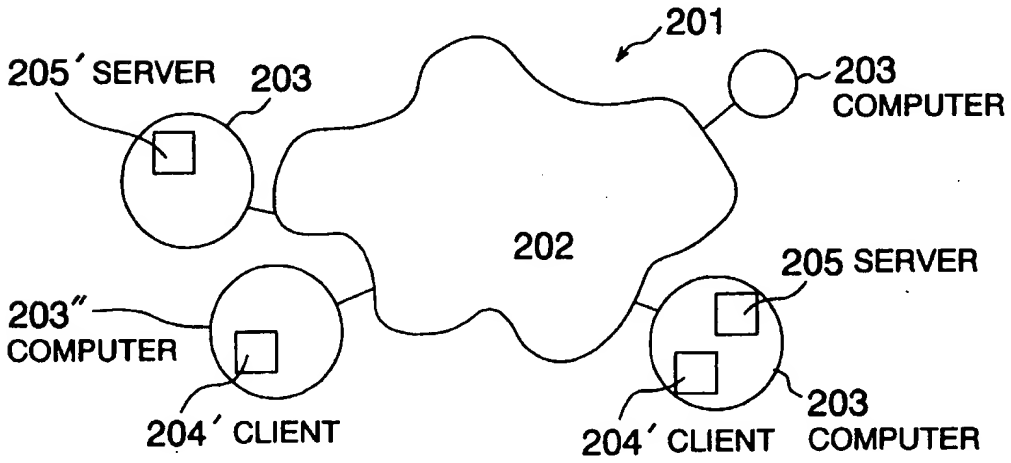

201
205' SERVER
203
203
COMPUTER
202
205 SERVER
203"
COMPUTER
204' CLIENT
204' CLIENT
203
COMPUTER

FIG. 3

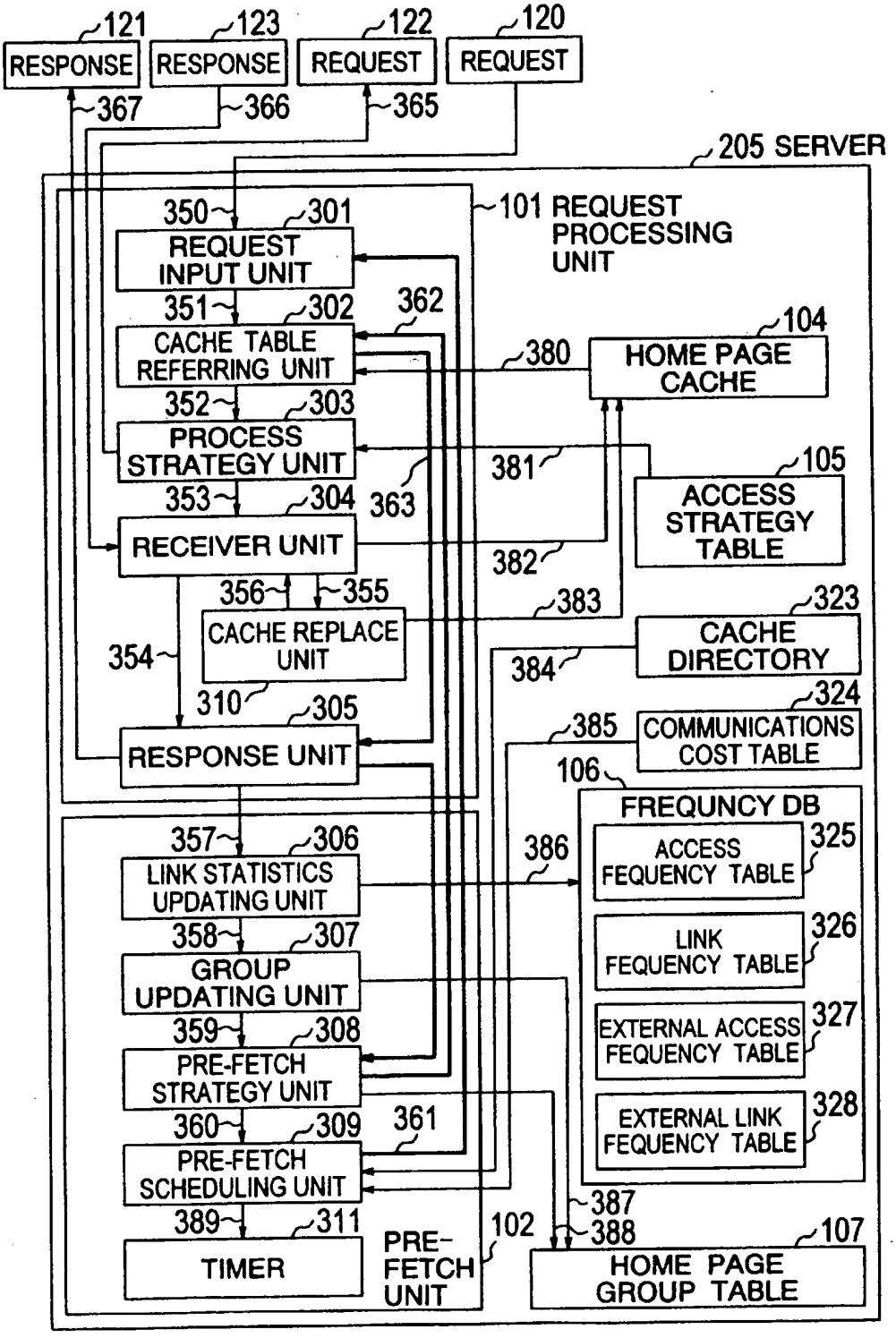

121
RESPONSE
123
RESPONSE
122
REQUEST
120
REQUEST
367
366
365
205 SERVER
101 REQUEST PROCESSING UNIT
350
301
REQUEST INPUT UNIT
351
302
362
CACHE TABLE REFERRING UNIT
380
104
HOME PAGE CACHE
352
303
PROCESS STRATEGY UNIT
381
105
ACCESS STRATEGY TABLE
353
304
363
RECEIVER UNIT
382
356
355
383
323
CACHE DIRECTORY
354
CACHE REPLACE UNIT
384
310
305
324
385
COMMUNICATIONS COST TABLE
RESPONSE UNIT
106
FREQUNCY DB
357
306
386
325
ACCESS FEQUENCY TABLE
LINK STATISTICS UPDATING UNIT
358
307
326
LINK FEQUENCY TABLE
GROUP UPDATING UNIT
359
308
327
EXTERNAL ACCESS FEQUENCY TABLE
PRE-FETCH STRATEGY UNIT
360
309
361
328
EXTERNAL LINK FEQUENCY TABLE
PRE-FETCH SCHEDULING UNIT
387
389
311
102
388
107
TIMER
PRE-FETCH UNIT
HOME PAGE GROUP TABLE

FIG. 4

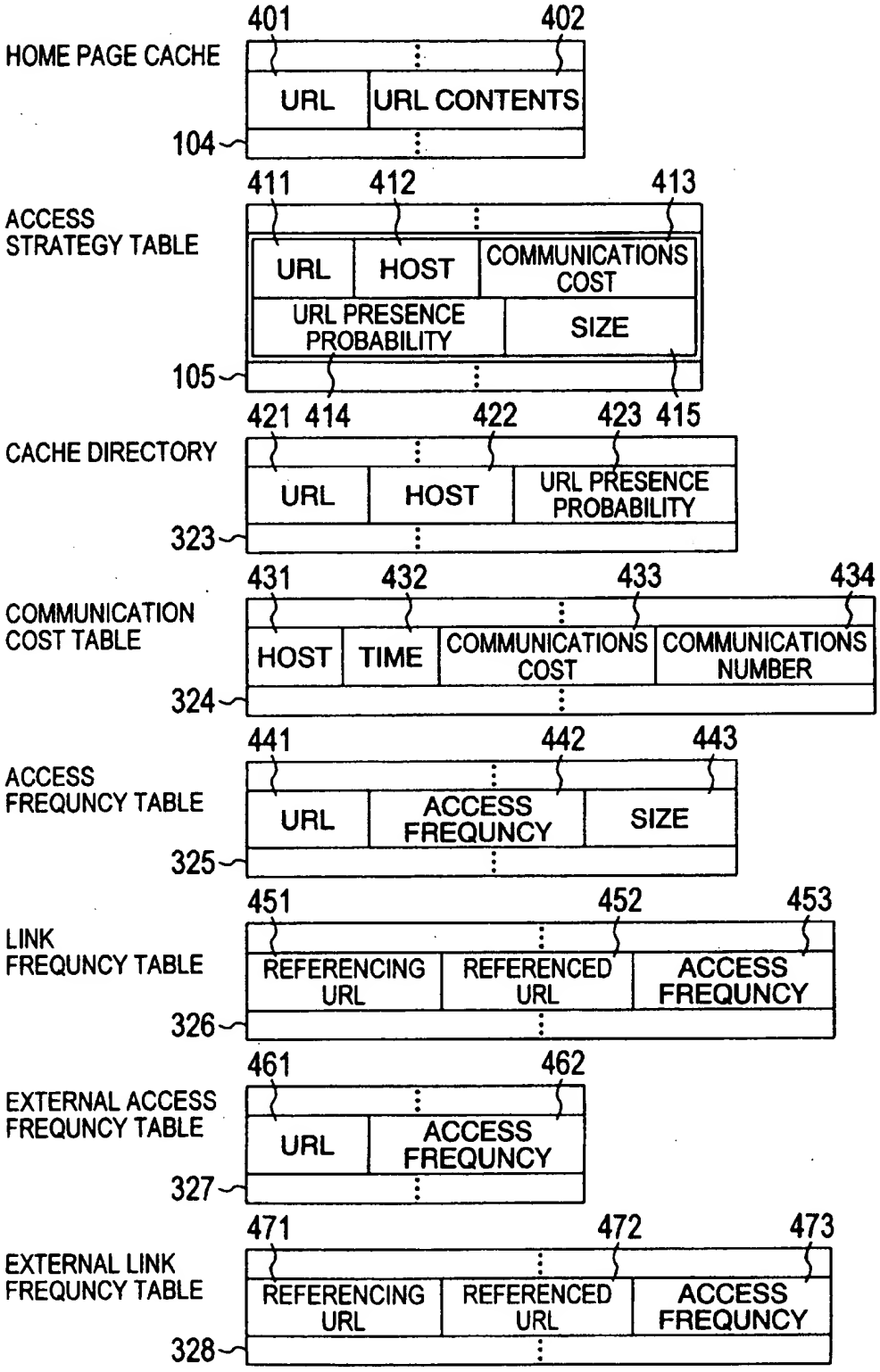

HOME PAGE CACHE
401
402
URL
URL CONTENTS
104
ACCESS STRATEGY TABLE
411
412
413
URL
HOST
COMMUNICATIONS COST
URL PRESENCE PROBABILITY
SIZE
105
CACHE DIRECTORY
421
414
422
423
415
URL
HOST
URL PRESENCE PROBABILITY
323
COMMUNICATION COST TABLE
431
432
433
434
HOST
TIME
COMMUNICATIONS COST
COMMUNICATIONS NUMBER
324
ACCESS FREQUNCY TABLE
441
442
443
URL
ACCESS FREQUNCY
SIZE
325
LINK FREQUNCY TABLE
451
452
453
REFERENCING URL
REFERENCED URL
ACCESS FREQUNCY
326
EXTERNAL ACCESS FREQUNCY TABLE
461
462
URL
ACCESS FREQUNCY
327
EXTERNAL LINK FREQUNCY TABLE
471
472
473
REFERENCING URL
REFERENCED URL
ACCESS FREQUNCY
328

FIG. 5

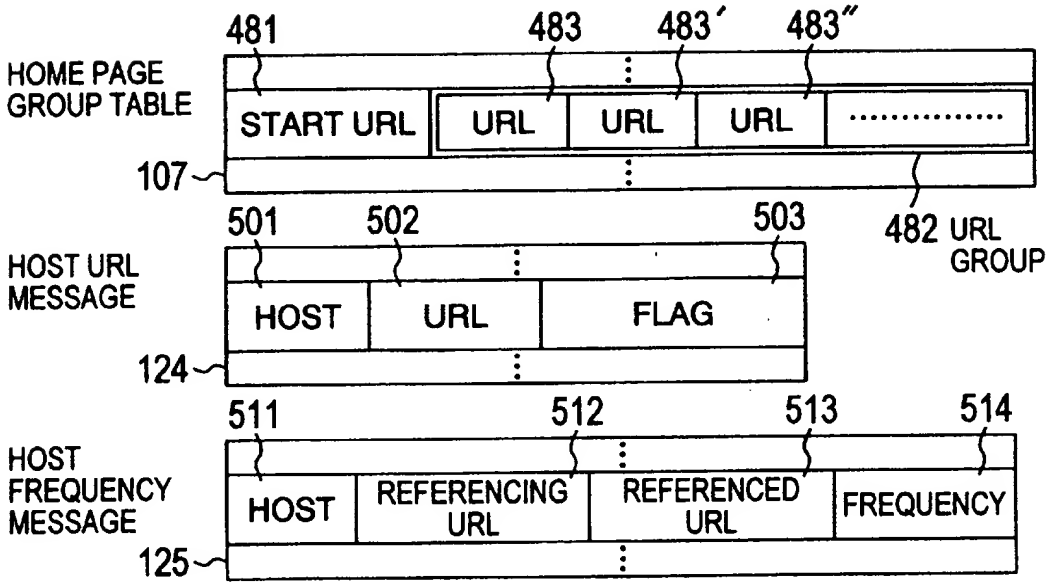

481
483
483′
483″
HOME PAGE GROUP TABLE
START URL
URL
URL
URL
.....
107
482 URL GROUP
501
502
503
HOST URL MESSAGE
HOST
URL
FLAG
124
511
512
513
514
HOST FREQUENCY MESSAGE
HOST
REFERENCING URL
REFERENCED URL
FREQUENCY
125

FIG. 6

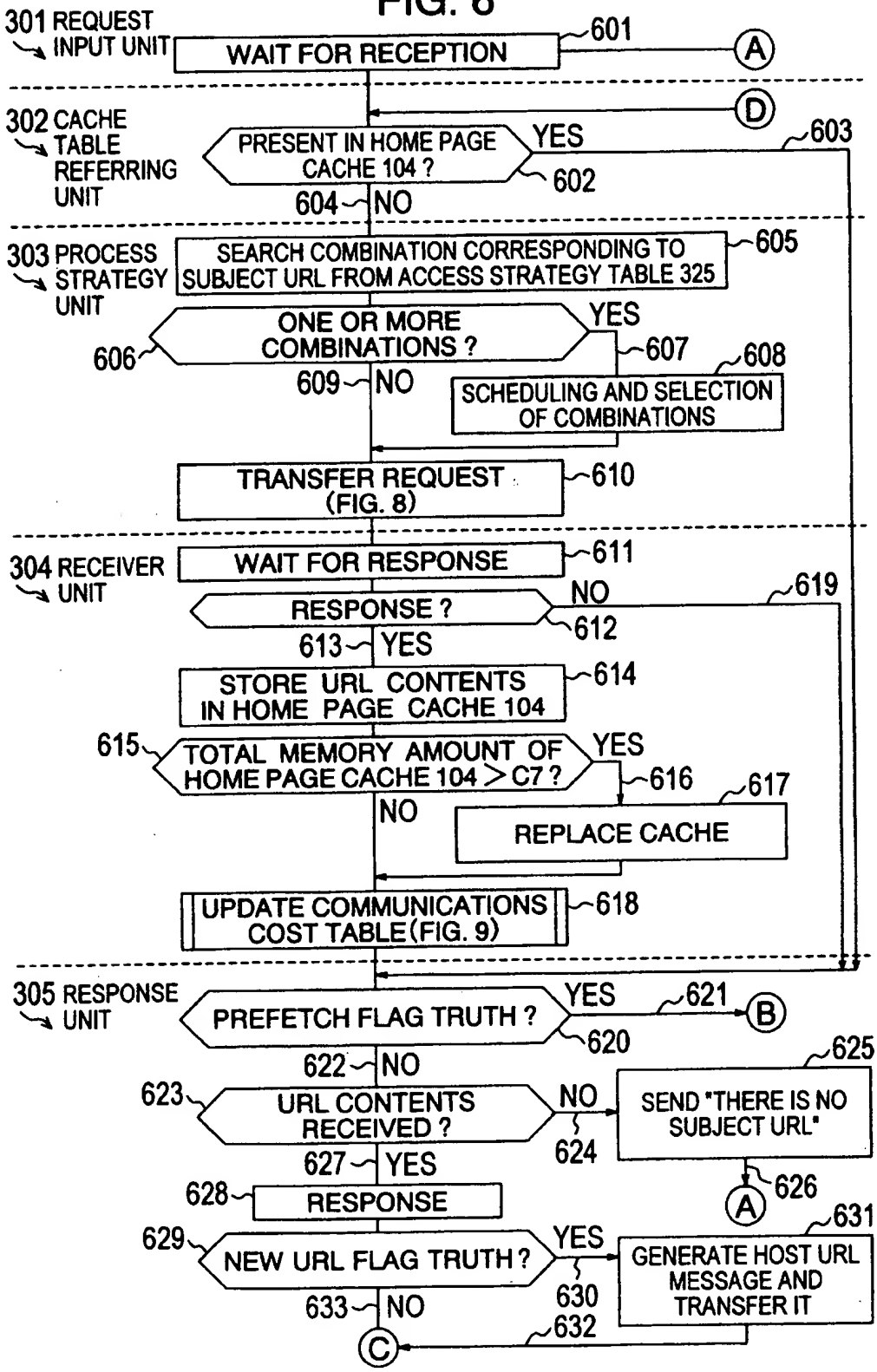

301 REQUEST INPUT UNIT
WAIT FOR RECEPTION
601
A
D
302 CACHE TABLE REFERRING UNIT
PRESENT IN HOME PAGE CACHE 104 ?
YES
603
602
604
NO
303 PROCESS STRATEGY UNIT
SEARCH COMBINATION CORRESPONDING TO SUBJECT URL FROM ACCESS STRATEGY TABLE 325
605
ONE OR MORE COMBINATIONS ?
YES
606
607
608
609
NO
SCHEDULING AND SELECTION OF COMBINATIONS
TRANSFER REQUEST (FIG. 8)
610
304 RECEIVER UNIT
WAIT FOR RESPONSE
611
RESPONSE ?
NO
619
612
613
YES
STORE URL CONTENTS IN HOME PAGE CACHE 104
614
615
TOTAL MEMORY AMOUNT OF HOME PAGE CACHE 104 > C7 ?
YES
616
617
NO
REPLACE CACHE
UPDATE COMMUNICATIONS COST TABLE (FIG. 9)
618
305 RESPONSE UNIT
PREFETCH FLAG TRUTH ?
YES
621
B
620
622
NO
625
623
URL CONTENTS RECEIVED ?
NO
SEND "THERE IS NO SUBJECT URL"
624
627
YES
626
A
628
RESPONSE
631
629
NEW URL FLAG TRUTH ?
YES
GENERATE HOST URL MESSAGE AND TRANSFER IT
630
633
NO
632
C

FIG. 7

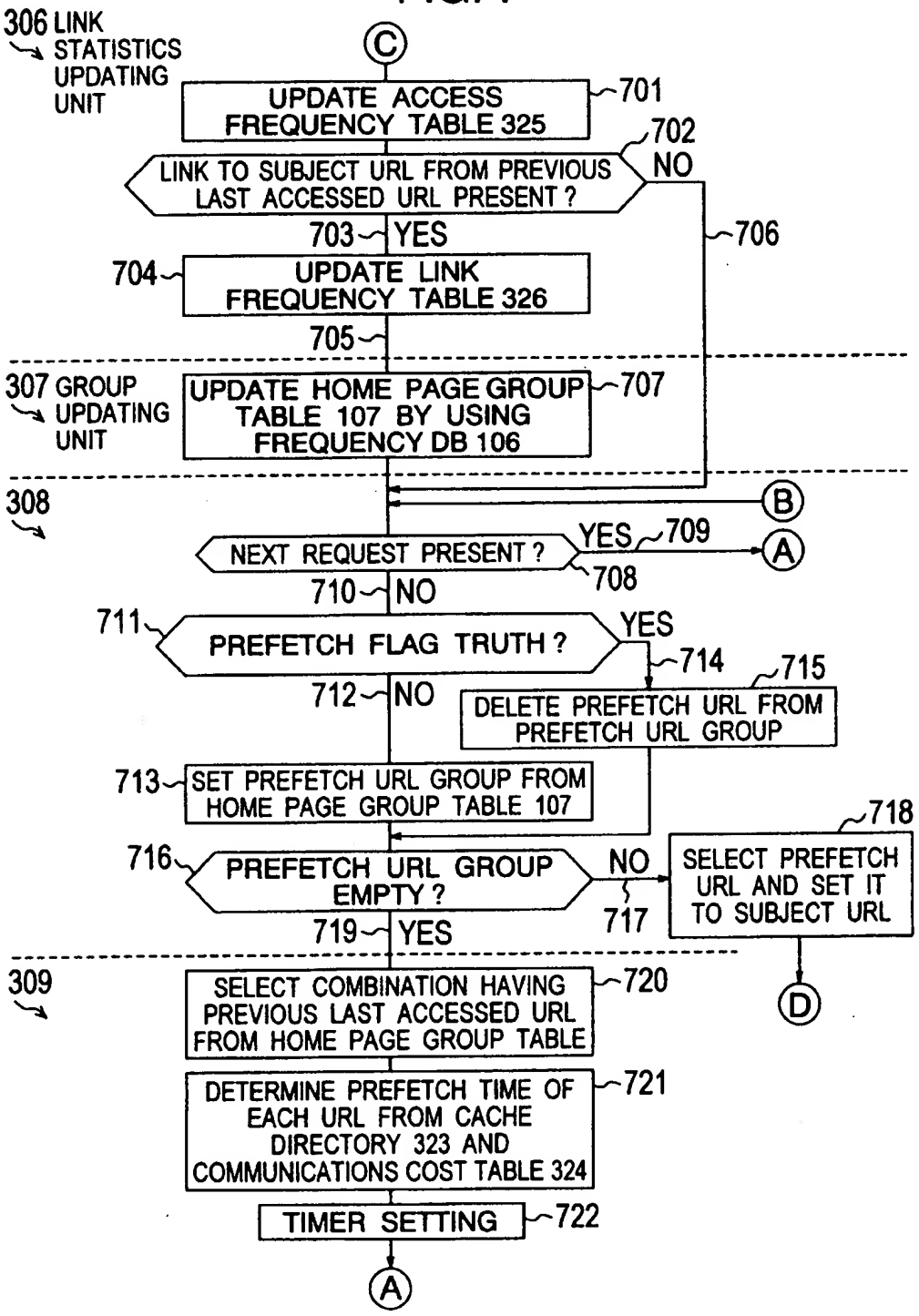

306 LINK STATISTICS UPDATING UNIT
C
UPDATE ACCESS FREQUENCY TABLE 325
701
LINK TO SUBJECT URL FROM PREVIOUS LAST ACCESSED URL PRESENT ?
702
NO
706
703
YES
704
UPDATE LINK FREQUENCY TABLE 326
705
307 GROUP UPDATING UNIT
UPDATE HOME PAGE GROUP TABLE 107 BY USING FREQUENCY DB 106
707
308
B
NEXT REQUEST PRESENT ?
YES
709
A
708
710
NO
711
PREFETCH FLAG TRUTH ?
YES
714
715
712
NO
DELETE PREFETCH URL FROM PREFETCH URL GROUP
713
SET PREFETCH URL GROUP FROM HOME PAGE GROUP TABLE 107
718
716
PREFETCH URL GROUP EMPTY ?
NO
SELECT PREFETCH URL AND SET IT TO SUBJECT URL
717
719
YES
D
309
SELECT COMBINATION HAVING PREVIOUS LAST ACCESSED URL FROM HOME PAGE GROUP TABLE
720
DETERMINE PREFETCH TIME OF EACH URL FROM CACHE DIRECTORY 323 AND COMMUNICATIONS COST TABLE 324
721
TIMER SETTING
722
A

FIG. 8A

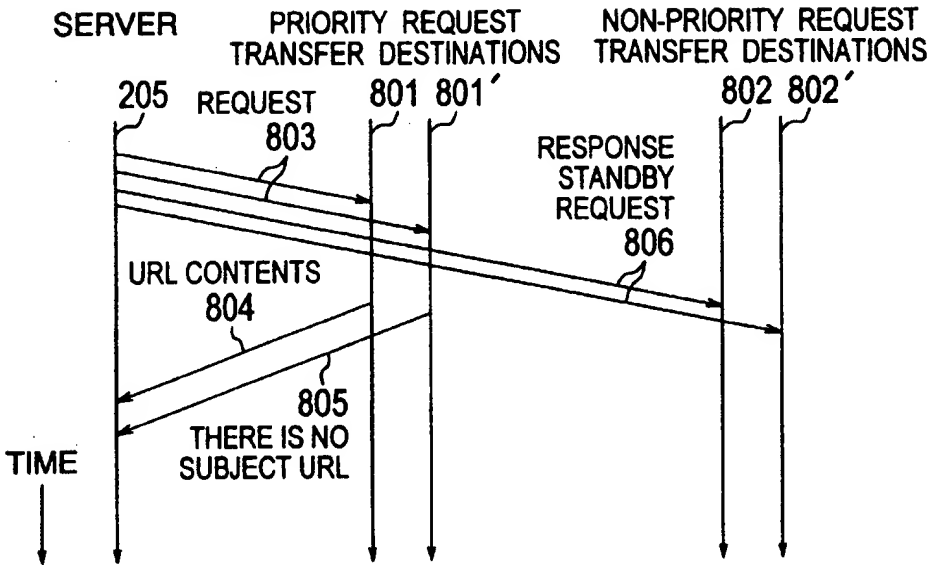

SERVER
PRIORITY REQUEST TRANSFER DESTINATIONS
NON-PRIORITY REQUEST TRANSFER DESTINATIONS
205
REQUEST 803
801 801′
802 802′
RESPONSE STANDBY REQUEST 806
URL CONTENTS 804
805 THERE IS NO SUBJECT URL
TIME

FIG. 8B

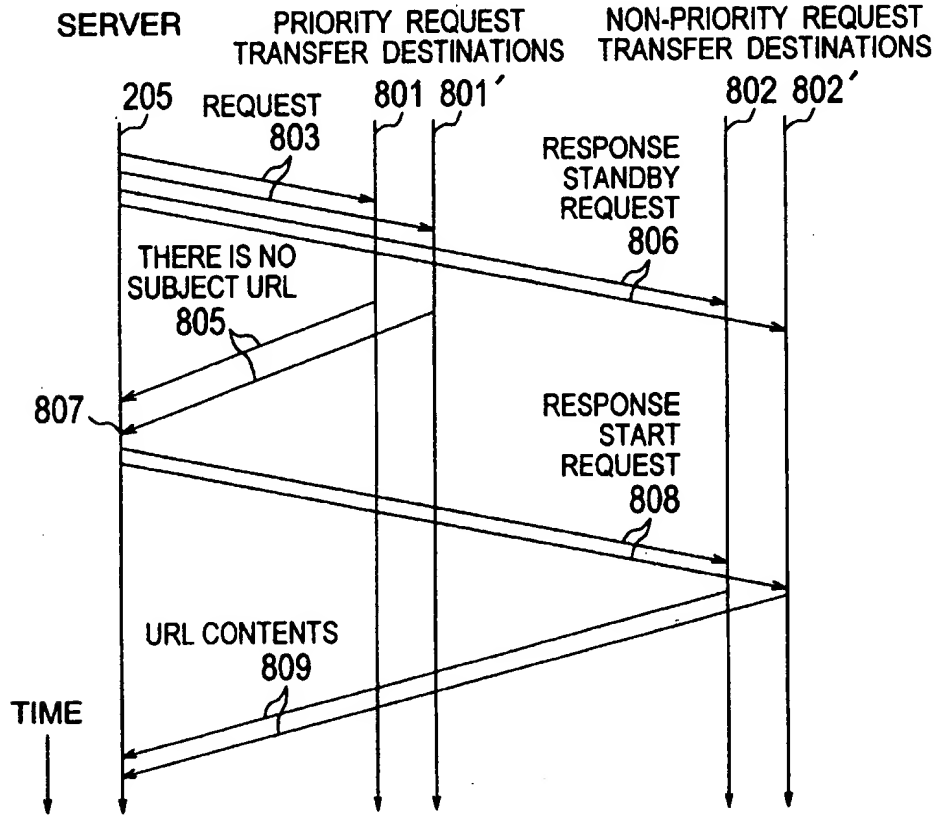
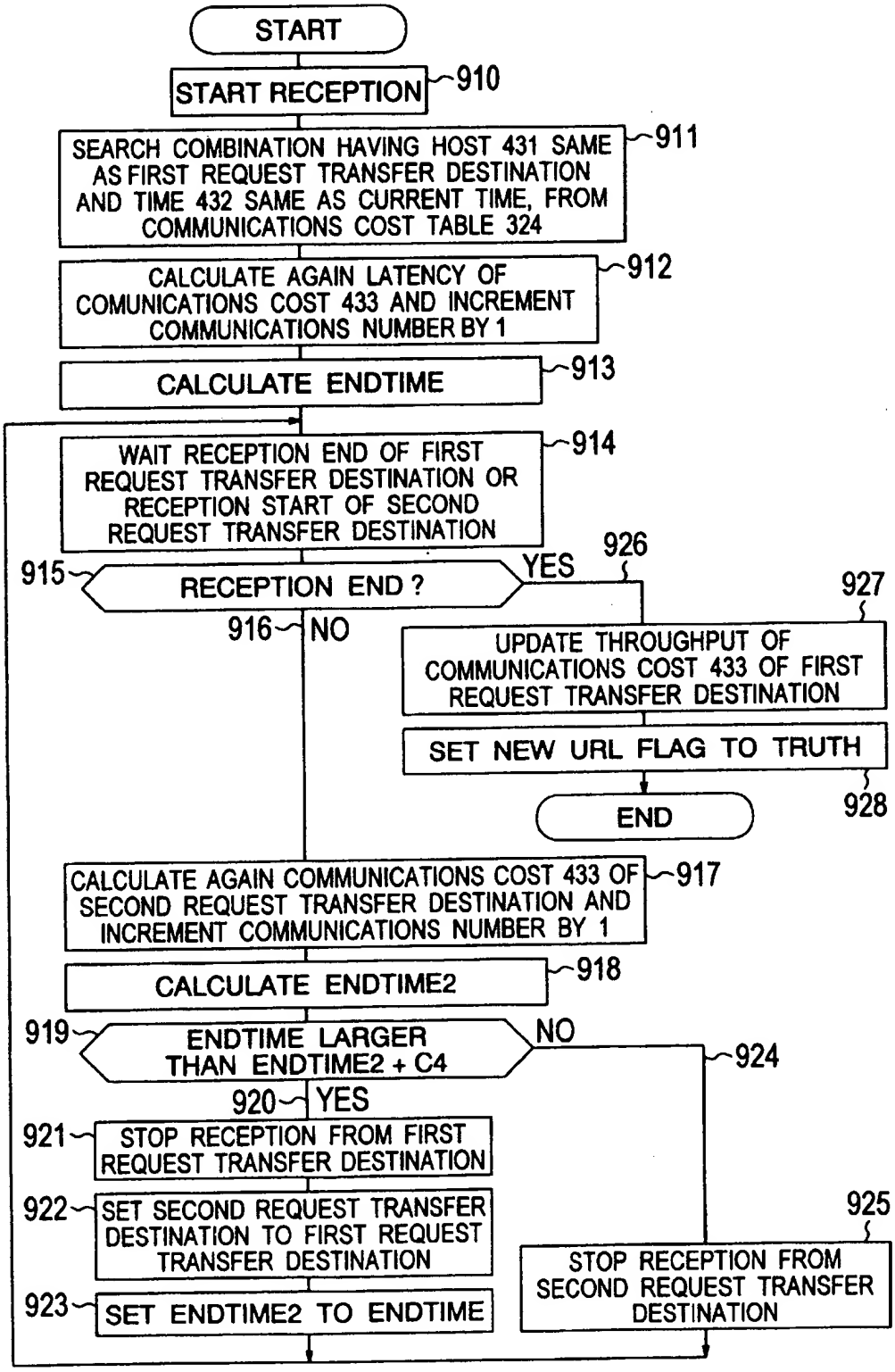

SERVER
PRIORITY REQUEST TRANSFER DESTINATIONS
NON-PRIORITY REQUEST TRANSFER DESTINATIONS
205
REQUEST 803
801 801′
802 802′
RESPONSE STANDBY REQUEST 806
THERE IS NO SUBJECT URL 805
807
RESPONSE START REQUEST 808
URL CONTENTS 809
TIME

FIG. 9


START
START RECEPTION
910
SEARCH COMBINATION HAVING HOST 431 SAME AS FIRST REQUEST TRANSFER DESTINATION AND TIME 432 SAME AS CURRENT TIME, FROM COMMUNICATIONS COST TABLE 324
911
CALCULATE AGAIN LATENCY OF COMUNICATIONS COST 433 AND INCREMENT COMMUNICATIONS NUMBER BY 1
912
CALCULATE ENDTIME
913
WAIT RECEPTION END OF FIRST REQUEST TRANSFER DESTINATION OR RECEPTION START OF SECOND REQUEST TRANSFER DESTINATION
914
915
RECEPTION END ?
YES
926
927
916
NO
UPDATE THROUGHPUT OF COMMUNICATIONS COST 433 OF FIRST REQUEST TRANSFER DESTINATION
SET NEW URL FLAG TO TRUTH
928
END
CALCULATE AGAIN COMMUNICATIONS COST 433 OF SECOND REQUEST TRANSFER DESTINATION AND INCREMENT COMMUNICATIONS NUMBER BY 1
917
CALCULATE ENDTIME2
918
919
ENDTIME LARGER THAN ENDTIME2 + C4
NO
924
920
YES
921
STOP RECEPTION FROM FIRST REQUEST TRANSFER DESTINATION
922
SET SECOND REQUEST TRANSFER DESTINATION TO FIRST REQUEST TRANSFER DESTINATION
923
SET ENDTIME2 TO ENDTIME
925
STOP RECEPTION FROM SECOND REQUEST TRANSFER DESTINATION

METHOD AND SYSTEM FOR MANAGING DISTRIBUTED DATA

BACKGROUND OF THE INVENTION

The present invention relates to a computer system, and more particularly to a method and system for managing distributed data, suitable particularly for the world wide web (WWW), in which a plurality of computers interconnected by a network distribute, share and exchange data in an information system.

First, several terms used in the following description will be explained.

An information system such as WWW and anonymous FTP on the Internet is configured as a "client-server system" which is one type of distributed computer systems. In the client-server system, the processes in the whole system are classified into two parts. The first part is executed by a program (hereinafter called a process) called a "server", and the second part is executed by processes called "clients". A client in the information system generally runs on a computer operated by a home user or a company user. The server in the information system stores information to be supplied to clients. The client in the information system stores new information in the server or requests information from the server.

It is common in a computer system that the same information is temporarily copied to a plurality of sites in order to access the information at high speed or increase a possibility of accessibility. Such a copy is discriminably called hint, cache, replica, stash and the like (refer to a document "Distributed Systems (1st ed.) compiled by Sape Mullender, pp. 13–15, ACM press, 1989). In the following, these copies are collectively called a "cache". To make a cache is called "cache".

A WWW server in WWW stores information to be serviced, in a unit called a "home page". Each home page has a name called URL (abbreviation for uniform resource locator). URL is a character string capable of designating a protocol used in WWW, a host name of a computer as an information source, and specific data in the information source. For example, "http://www.hitachi.co.jp/index.html" is a URL.

Generally, each URL is in correspondence with a collection of data including character and image data of a home page. In the following, a collection of such data is called "URL corresponding information" or "URL contents". A second URL contained in the first URL corresponding information is called a "hyper text link (or simply link)". That the first URL corresponding information contains a second URL is hereinafter described as "there is a link from the first URL to the second URL".

The techniques (hereinafter called prior example 1) used by WWW will be explained in the following.

A user of a WWW client supplies a URL of the home page to be accessed, to the WWW server. In the first process type between a WWW server and client, the WWW client requests the WWW server designated by the second element of URL to transmit the home page of URL. In response to this request, the WWW server supplies the home page to the WWW client.

In the second process type, instead of requesting the WWW server designated by the second element of URL supplied from the user, the WWW client requests a second server called a "proxy server" to transmit the home page. The second server acquires the home page of URL from the first WWW server or requests another proxy server to acquire the URL corresponding information. At the repetitive stage of proxy server requests, these proxy servers have parent-child relationships. Proxy servers having a parent-child relationship are described, for example, in a document "A Hierarchical Internet Object Cache" by A. Chankhunthod, et.al., 1996 USENIX Technical Conference, pp. 153–163, 1996.

A WWW client and proxy server can have caches. A cache of a client stores home pages the client acquired in the past, and can be used only by this client. A cache of a proxy server stores home pages acquired by the proxy server in response to a request from one or more clients, a request from another or more other servers, or a request from both, and can be shared by the clients using this proxy server or by this proxy server itself.

The Network News System (hereinafter called prior example 2) is described, for example, in a document "Network News Transfer Protocol: A proposed Standard for the Stream-Based Transmission of News" by B. Kantor, et.al., Network Working Group RFC-977. This system is configured by one or more servers. Generally, a user selects one of the servers by using its client. The information unit in the Network News System is called "news". Generally, a user supplies news to the server by using its client, and acquires news from the server. As the user supplies news to a first server, the first server sends a copy of the news to a second server, and the second server supplies a copy of the news to another server, and so on. Finally, the copy of the news is supplied to all the servers.

Next, the global area name service, Domain Name System (hereinafter called prior example 3, abbreviated as DNS) will be explained. DNS is described, for example, in a document "Domain Names-Implementation and Specification" by P. Mockapetris, Network Working Group RFC-1035, particularly in the second section thereof. DNS has a correspondence mainly between a symbolic host name and host related information (IP address and mail address). A plurality of DNS servers have a tree structure. A request from a client is processed by tracing the tree structure and transferring the request to a plurality of servers. A resolver which is a DNS client requests the host related information corresponding to a host name to one of the servers. This server returns the host related information corresponding to the host name back to the client, or transfers the request to a parent server of this server (a DNS server nearer to the root of the DSN server tree structure from this server). The parent server grasps which of its child servers have what host related information. Therefore, after the request is transferred to the root of the tree structure, the request is then transferred downward the tree structure to the DNS server capable of processing the request. The request finally reaches the DNS server having the host related information from which the host related information is returned to the client, or alternatively a failure is returned back to the client if every DNS server cannot supply the host related information corresponding to the host name while the request is transferred downward the tree structure.

A method (hereinafter called prior example 4) is also known in which a space of caches is shared by a plurality of computers in a distributed file system of a local area network (LAN). According to a document "Cooperative Caching: Using Remote Client Memory to Improve File System Performance" by Michael Dahlin, et.al., First USENIX Symposium on Operating Systems Design and Implementation, pp. 267–280, 1994, a client first requests for a file block to a server called a "manager". The manager

grasps which file block is stored in what computer. The manager informs the client of the computer which stores the file block, or transfers the request of the client to the computer. Similar methods are known as described in a document "Implementing Global Memory Management in an Workstation Cluster" by M. J. Feeley, et.al., ACM 15th Symposium on Operating Systems Principles, pp. 201–212, 1995, or in a document "Efficient Cooperative Caching Using Hints" by P. Sarkar, et.al., Second USENIX Symposium on Operating Systems Design and Implementation, pp. 35–46, 1996. A plurality of managers can be provided. However, a correspondence between file blocks and managers is prefixed and is known by all clients and servers. This correspondence does not change during system running.

Techniques used by computers called cache-coherent non-uniform memory access (CC-NUMA) and cache only memory access (COMA) will be explained by using following prior examples 5 and 6. The CC-NUMA computer or COMA computer has a mechanism of maintaining coherency between memory fragments (cache lines) cached near at a number of processors. The following two methods are known in particular.

With the first method (hereinafter called prior example 5), a processor or data called a "home" corresponding to the "manager" grasps which memory fragment is cached to what processor. This first method is used in a system described, for example, in a document "The Stanford FLASH Multiprocessor" by Jeffrey Kuskin, et.al., Proceedings of the 21th Annual International Symposium on Computer Architecture, pp. 302–314, ACM, 1994.

With the second embodiment (hereinafter called prior example 6), some restrictions are imposed on the formation, deletion and communications of caches, to thereby ensure identification and coherence of caches during a predetermined number of communications (generally including multicast or broadcast). This second method is used in a system described, for example, in a document "The Data Diffusion Machine with a Scalable Point-to-Point Network" by Henk L. Muller, et.al., Technical Report CSTR-93-17, Department of Computer Science, University of Bristol, October 1993.

The current communications performance of the Internet is much more slower than the speed users desire, and has various unstable factors. Because of rocketing spread of the WWW, a number of wide area networks (WAN) are congested. While high speed backbone communications lines are being increased and enhanced day after day, users at homes connect to the Internet via communication lines much slower than a general LAN. The number of users of WWW servers and the Internet is increasing even at present. According to certain statistics as of January 1996, the number of computers connecting the Internet in the world is nine million, and increasing by twofold in less than six months.

These circumstances make Internet communications lines irregular and unstable. "Irregular" means congestions of various communications lines. For example, each communications line has a different throughput (communications data amount per unit time) and a different latency (communications delay time). "Unstable" means that the throughput and latency of a communications line change from time to time and at worst the communications becomes impossible. For example, the congestion degree of a communications line changes with a time zone and a day of the week, or a routing pattern changes because of some enhanced communications line so that another communications line becomes congested or becomes free of congestion.

Under such circumstances, it is required to shorten the time from when a user issues a request to a client to when the request is satisfied, in order for the information system to provide services more comfortable to users. The following issues (1) to (5) regarding such user requirements will be explained.

(1) Under the conditions that some communications line is unstable, a client request may not be processed at high speed even if another communications line operates normally.

A WWW client and a WWW proxy server of the prior example 1 communicate with a specific WWW server and a specific proxy server designated by URLs. Therefore, if the communications line to the server (or proxy server) is congested or interrupted, it takes a long time for the client (or proxy server) to access a home page, or the home page cannot be accessed, even if another communications line operates normally. From the same reasons, even if some communications line is enhanced to have a high speed, it has been difficult to enjoy the high performance of this line. These problems are also true for the prior example 2 to 5, because communications between a client and a server or between a server and another server is performed by prefixed partners, similar to the prior example 1. The prior example 6 pertains to the techniques to be used by a single computer or by a plurality of computers physically in tight contact with each other, in which multicast or broadcast is assumed to be normally operative. Therefore, it is difficult to widely apply these techniques to the LAN and WAN environments. From the same reasons, if a particular communications line is congested or interrupted, it takes a long time to obtain a cache line or the cache line cannot be obtained, even if another communications line operates normally.

(2) A usage factor of caches cannot be improved under the unstable state of communications lines.

In the prior examples 1, 2, 3 and 5, a plurality of caches at a plurality of servers (or clients and servers) have a hierarchical structure. A request issued by a client is processed by a specific first server responsible for this process. This request is either directly transmitted to the first server or sequentially transferred to one or more second servers present between the first server and the client (in this case, one or more second servers are determined definitely in accordance with the contents of the request). In the former case, the request can use the cache of only the first server. In the latter case, the request can use only cache or caches of one or more second servers. Namely, the request cannot use caches of servers other than the first and second servers so that the usage factor of caches is small. For example, in the prior example 1, WWW proxy servers have a parent-child relationship. It is assumed here that there are three proxy servers A, B and C, the server A being a parent and the servers B and C being children of the server A. In this case, although B can use the cache of A by transferring a request to A, B cannot use the cache of C. In the prior example 2, each news is copied to all servers requesting it. Therefore, each child server does not use at all the cache of another child server so that each child server is required to prepare a large scale secondary memory. The prior example 4 assumes the LAN environment in which in order to improve the usage factor of caches, it is controlled to make copies of one data set be as small as in a plurality of caches. However, if a communications line is unstable, a client request may not reach such a small number of caches, resulting in a low cache usage factor. In the prior example 6, of a plurality of caches, a cache which can be acquired in shortest time is selected. For this selection, multicast or broadcast is per-

formed. Therefore, this technique is difficult to be widely applied to the LAN and WAN environments whose communications lines may become unstable.

(3) A method of exchanging among a plurality of servers a use frequency of data reference relationships, has not been applied.

As typical in WWW home pages and hyper text links, data provided by an information system has reference relationships. Of these reference relationships, there is in many cases a reference relationship frequently accessed (reference relationship having a high use frequency). This use frequency of reference relationships can be used for prefetch. If a series of information linked by reference relationships frequently accessed is prefetched before a client issues a request, this series of information can be processed at high speed when a client issues a request, even if communications lines are congested. If there are a plurality of servers, information on the use frequency of reference relationships can be made more reliable if a plurality of servers collect and exchange the information and summarize it, than if the information is collected independently by each server. However, the prior examples 1 to 6 have not the method of exchanging among a plurality of servers a use frequency of data reference relationships. Therefore, reliability of the information on a use frequency of reference relationships is limited and the effect of prefetch is also limited.

(4) It has been a possibility of discarding an important cache when caches are replaced, because irregular and unstable communications lines have not been taken into consideration.

In the prior examples 1, 2 and 3, each server (or client) replaces caches by referring to a use history or the like. Since the prior example 4 assumes the LAN environment, the states of communications lines are not taken into consideration when the priority level of caches is determined to replace them. The prior examples 5 and 6 also assume one computer or a plurality of computers physically in tight contact with each other. Therefore, the states of communications lines are not taken into consideration when the priority level of caches is determined to replace them. In all the prior examples 1 to 6, therefore, a cache unable to be obtained under the current communications conditions or a cache taking a long time to obtain, may be discarded.

(5) While a first server accepts a request from one or more second servers, if the second servers use the first server limitlessly, the first server may be overloaded. A countermeasure for this has not been applied to the prior examples 1 to 6. It is therefore difficult for a plurality of servers to pass requests from users other than a predetermined society of a plurality of users. In the prior examples 1, 2 and 3, since the client-server system is adopted, if the server rejects a client request, this request cannot be processed. In the prior examples 4, 5 and 6, since servers of a limited number pass requests, an overload of the first server does not become a problem.

SUMMARY OF THE INVENTION

An object of the invention is to solve the above issues (1) to (5) and provide services of an information system more comfortable to users, by providing a plurality of caches to a plurality of servers.

In order to solve the issue (1), the following three techniques are provided.

(i) In acquiring data necessary for a first server (hereinafter called necessary data), this necessary data may be cached at two or more second servers among a plurality of servers using the invention method. In this case, the first server has past communications history between first and second servers, and in accordance with the communications history, selects one or more third servers from the second servers, and requests the third servers to transmit the necessary data. In accordance with the communications history, the first server can select the third server which can be expected to acquire the necessary data at high speed at present. It is therefore possible to process a client request by selectively using a communications line which can perform communications at high speed.

(ii) The first server selects two or more third servers from which the necessary data is acquired, and a request is transmitted at the same time to a plurality of servers so as to process the client request at high speed. However, in this case, responses from the plurality of servers start generally at the same time and the communications speed may be lowered. In order to avoid this, of two or more second servers, one or more third servers are requested to transmit the necessary data at once, whereas another or more fourth servers are requested to hold the transmission of the necessary data. When it is judged that it takes a long time for the third server to transmit the necessary data or that the necessary data cannot be transmitted, the stand-by fourth server immediately transmits the necessary data. It is therefore possible to process the client request at high speed even if the communications state changes.

(iii) The first server has past communications history (communications history with time) during a first time at a second time interval between the first server and one or more other servers. This communications history with time allows the first server to select a suitable time for communications, even if the communications line state changes periodically (e.g., if a communications line congested in the day time but not congested in the night time is used). With these three techniques (i) to (iii) of this invention, a communications line capable of communications at high speed can be selectively used, or a communications line is selectively used in a time zone when communications can be performed at high speed, so that the client request can be processed at high speed even under the conditions of irregular and unstable communications lines.

In order to solve the issue (2), the first server transmits part or the whole of a list of caches possessed by the first server to one or more second servers. In this case, one or more second servers are selected in accordance with the history of communications line state. The list of caches of the first server can be transmitted to the second server capable of performing communications at high speed. If the communications line between the first and second servers is normal, the second server can use the cache of the first server even if other communications lines are unstable.

In order to solve the issue (3), for the case wherein there is a high possibility that after a client requests for first data, the client requests one or more sets of second data, the first server stores a correspondence (reference information) between first and second data, and transmits this reference information to one or more second servers. With this means, each server in a plurality of servers of this invention exchanges the reference information with another server to improve the reliability of use frequency of the reference often referred. Therefore, a series of data frequently requested can be provided to the client at high speed.

If the reference information is prefetched, a number of data sets can be requested at the same time. In this invention,

a plurality of servers prefetch data sets hierarchically in a pipe line manner so that effective prefetch can be performed without a bottle neck at a particular server or an overload on a particular network.

In order to solve the issue (4), the first server determines a priority level of each cache by using the history of the communications line state, in accordance with an access frequency of information unit, a last accessed time, an average throughput, an average latency, and the contents of caches of other servers. In accordance with the determined priority level, caches are replaced. It is therefore possible to prevent a cache from being discarded, which cache cannot be acquired under the current communications conditions or takes a long time to be acquired.

In order to solve the issue (5), the first server has means for limiting the transmission amount of necessary data from one or more second servers to a predetermined amount during a predetermined time, or to a predetermined number of transmissions during a predetermined time. Each of the second servers requests two or more third servers including the first server to transmit the necessary data. It is therefore possible to prevent an overload of the first server to be caused by the second servers and not to obstruct the processes of the second servers. With this means of the invention, even if one server among a plurality of servers rejects a request, another server can process the request.

With the above means, a time from when a user issues a request to a client to when the request is processed can be shortened even under the conditions of irregular and unstable communications lines of the Internet. A request by a client can be processed as much as possible even if a particular communications line is interrupted. In this manner, services of a communication system more comfortable to users can be provided.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram showing the outline of the internal configuration of a server according to an embodiment.

FIG. **2** is a diagram showing a distributed computer system according to an embodiment.

FIG. **3** is a block diagram showing the internal structure of a server.

FIG. **4** is a diagram showing a data structure used by the server **205**.

FIG. **5** is a diagram showing the data structure used by the server **205**.

FIG. **6** is a flow chart illustrating processes to be executed by the server.

FIG. **7** is a flow chart illustrating processes to be executed by the server.

FIGS. **8**A and **8**B are diagrams illustrating a protocol to be used by servers for alleviating congestion.

FIG. **9** is a flow chart illustrating a process of updating a communications cost table.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Embodiments of the invention will be described with reference to the accompanying drawings.

<Overall Structure>

A computer system of this embodiment uses a distributed data management method of storing data from another information system or from a user in one or more distributed caches. FIG. **2** shows the overall structure of this embodiment. The computer system **201** of this embodiment is a distributed computer system having one or more computers **203**, **203'**, **203"**, . . . interconnected by a network **202**.

The network **202** may be LAN often used by the whole or part of a community (company, school or like communities) or may be the whole or part of WAN interconnecting a plurality of geographically distributed sites. The network **202** may also be a computer coupled network or a processor coupled network of parallel computers.

Each of the computers **203**, **203'**, **203"**, . . . runs one or more clients, one or more servers, or both. The computer system **201** of this embodiment has at least one server **205** and at least one client **204**. In the following, a server group contained in the computer system **201** of this embodiment is called a "present system" where applicable. Each of the computers **203**, **203'**, **203"**, . . . may be any computer such as a personal computer, a workstation, a parallel computer, and a main frame computer. Each of the computers **203**, **203'**, **203"**, . . . running a corresponding one of servers **205**, **205'**, **205"**, . . . has a function of communicating with a corresponding one of clients **204**, **204'**, **204"**, . . . Namely, each of the computers **203**, **203'**, **203"**, . . . may be a computer terminal, a portable communications terminal (personal digital assistance PDA, hand-held personal computer HPC), a network computer or the like. The number and structure of the computers **203**, **203'**, **203"**, . . . clients **204**, **204'**, **204"**, . . . and servers **205**, **205'**, **205"**, . . . shown in FIG. **2** are only illustrative and are not intended to limit the scope of this invention.

A client **204** communicates with a server **205** by using various protocols, such as HTTP (abbreviation for Hypertext Transfer Protocol) and FTP (abbreviation for File Transfer Protocol). The client **204** may be a WWW browser with graphic user interface, a WWW browser only with text display, a WWW client bundled in a word processor or a spreadsheet software, or a WWW proxy server. These protocols used by the client **204** and these types of the client **204** are only illustrative and do not limit the scope of the invention.

The servers **205**, **205'**, **205"**, . . . obtain information to be supplied to clients from the information sources such as a WWW server, WWW proxy server, anonymous FTP server, and user, and store the information paired with URL. The servers **205**, **205'**, **205"**, . . . obtain new information by periodically (e.g. one per hour) connecting the information sources, by being notified of a generation of new information by the information sources, or by other methods. These information sources and information retrieval methods used by the servers **205**, **205'**, **205"**, . . . are only illustrative and do not limit the scope of the invention. The information stored in the servers **205**, **205'**, **205"**, . . . is added with URL and managed by using URL. Instead of URL, other identifiers may be used if they can designate the information stored in the servers. Use of URL is not intended to limit the scope of the invention. The embodiment of the invention is independent from the type of an operating system used by a client or server, the type of a network between servers or between servers and clients, the type of a network physical layer protocol and a network transport layer protocol, and whether the server and client are on the single computer or on different computers.

<Outline of Operation of Client and Server>

With reference to FIG. **1**, the outline of the operations of the client **204** and server **205** will be described. In FIG. **2**, the network **202** interconnecting the server **205**, client **204** and other servers **205'**, **205"**, . . . is not shown. Arrows in FIG. **1** indicate main data flows.

The server **205** has processing units and data structures. The processing units include a request processing unit **101**, a prefetch unit **102**, and an interserver message processing unit **103**. The data structures include a home page cache **104**, an access strategy table **105**, a frequency DB **106**, and a home page group table **107**.

The home page cache **104** of the server **205** is stored in a non-volatile storage (such as a magnetic hard disk and an optical disk which are described hereinafter as "secondary storage" where applicable), a volatile storage (such as a main memory and a cache memory which are described hereinafter as "main storage" where applicable), or both, of the computer **203** running the server **205**. The home page cache **104** is not required to be stored in one area of the non-volatile storage or volatile storage. For example, if the computer **203** running the server **205** has three magnetic hard disks, the home page cache **104** may be stored divisionally in some or all of the three magnetic hard disks. It is not essential but desirable to divisionally store the home page cache **104** in a plurality of magnetic hard disks, because the parallel access performance of the magnetic hard disks is improved and read/write of the home page cache **104** is speeded up. Caching information is an important role of the server **205**. It is therefore not essential but desirable to provide the server **205** with a storage sufficiently large for constituting a large capacity home page cache **104**.

In order for the client **204** to acquire first URL contents or first URL related information from the present system, a request **120** containing the first URL is transmitted to an optional first server **205** of the present system.

The request **120** is received by the request processing unit **101** which searches the home page cache **104** to check whether it contains the requested URL contents (**130**). If it contains, the home page cache **104** is accessed and the first URL contents are transmitted to the client **201** at a response **121**.

The access strategy table **105** stores therein records indicating which URL contents the other servers **205'**, **205"** of the present system have, and records indicating past communications speeds (e.g., throughput and latency) used by the first server **205**. If the first URL contents are not contained in the home page cache **104** of the first server **205**, the first server **205** refers the access strategy table **105** to determine the second servers **205'**, **205"**, . . . to which the request is transferred (**131**). The request **120** is transferred as a request **120** to one or more of the second servers **205'**, **205"**, . . . and to the information sources of the first URL contents. When the first URL contents are acquired as a response **121** from at least one of the second servers **205'**, **205"**, . . . and the information sources to which the request **122** was transferred, the first server **205** sends the acquired first URL contents to the client **204** as a response **121**. The requests **120** and **122** and responses **121** and **123** may be transmitted via the network **202** or by using the communications function of the computers **203**, **203'**, **203"**, . . .

When the first server **205** acquires the first URL contents as the response **123**, it adds the first URL contents to the home page cache **104** of the first server **205** (**133**). In this case, a host URL message **124** indicating an addition of the first URL contents is transmitted to some or all of the other servers **205'**, **205"**, . . . of the present system. When the first URL contents are added to the home page cache **104**, second URL contents may be deleted from the home page cache **104**. In this case, the host URL message **124** indicating a deletion of the second URL contents is transmitted to some or all of the other servers **205'**, **205"**, . . . of the present system.

The frequency DB **106** stores frequency information indicating which URL contents was referred at what time, as will be later described. The home page group table **107** stores therein a plurality of combinations of URL contents frequently requested consecutively. After the response **121**, the prefetch unit **102** of the first server **205** reflects upon the frequency DB a change in the frequency information caused by the request **120** (**135**). A change in the frequency information is transmitted to some or all of the other servers **205'**, **205"**, . . . of the present system, by transmitting a host frequency message **125** representative of the update contents of the frequency DB **106**. The prefetch unit **102** updates the home page group table **107** in accordance with the contents of the frequency DB **106** (**136**).

Next, the URL contents expected to be requested in the future are determined by referring the frequency DB **106** and home page group table **107** (**137** and **138**). If there are one or more sets of the expected URL contents, these sets are received by the request processing unit **101**, similar to the case of the request **120**.

When the inter-server message processing unit **103** receives a host URL message **124'** from the other servers **205'**, **205"**, of the present system, it renews the access strategy table **105** (**140**). When the unit **103** receives a host frequency message **125'** from the other servers **205'**, **205"**, of the present system, it renews the frequency DB **106** (**141**).

The outline of the operations of the client **204** and server **205** has been described above. The details thereof will be given hereinunder.

<Details of Internal Operation of Server>

FIG. **3** shows the internal structure of the server **205**. The request processing unit **101** is constituted of a request input unit **301**, a cache table referring unit **302**, a process strategy unit **303**, a receiver unit **304**, a response unit **305**, and a cache replacing unit **310**. The prefetch unit **102** is constituted of a link statistics updating unit **306**, a group updating unit **307**, a prefetch strategy unit **308**, a prefetch scheduling unit **309**, and a timer **311**. The inter-server message processing unit **103** is not shown in FIG. **3**.

The data structures of the server **205** are constituted of the home page cache **104**, the access strategy table **105**, the frequency DB **106** (constituted of an access frequency table **325**, a link frequency table **326**, an external access frequency table **327**, and an external link frequency table **328**), the home page group table **107**, a cache directory **323** and a communication cost table **324** both for configuring the access strategy table **105**, and a host table (not shown). These data structures may be stored in one or both of the main and secondary storages. In FIG. **3**, bold arrows indicate main control flows, and thin arrows indicate main data flows. In this embodiment, the necessary data as recited in the appended claims corresponds to the URL contents, the communications history as recited in the appended claims is stored in the access strategy table **105** and communications cost table **324**, the data storage probability is stored in the cache directory **323**, the communications history with time is stored in the frequency DB **106**, and the reference information is stored in the home page group table **107**. A list of data transferred between servers is transmitted in the form of a host URL message **124**.

Prior to the description of the internal processes of the server **205**, the outline of each data which a request is transferred.

As the host name, a symbolic host name or an IP address is used, or in some cases a combination of a symbolic host name or an IP address and a port number of TCP/IP or UDP/IP is used. Since the main objective of the host name

is to designate a communications partner server or client and establish a communications path, the host name may be described as desired so long at it satisfies such an objective. The IP address and port number are only illustrative and are not intended to limit the scope of the invention.

The cache directory **323** stores therein generally a plurality of combinations of a URL **421**, a host **422** and a URL presence probability. The URL **421** is a URL itself. The host **422** is a name of the server or a name of the host having an information source. The URL presence probability **423** is a probability of a presence of a home page designated by URL **421** in the computer designated by the host **422**.

The communication cost table **324** stores therein generally a plurality of combinations of a host **431**, a time **432**, a communications cost **423**, and a communication number **434**. The host **431** is a name of the server or a name of the host having an information source. The time **432** stores a time and a day of the week. The communications cost **433** is represented by throughput (transferable data amount per unit time) and latency (communications delay time) of communications with the host **431** at the structure will be described first. The detailed description of each data structure will be later described together with the internal processes of the server **205**. FIG. **4** shows the structures of the home page cache **104**, access strategy table **105**, cache directory **323**, communications cost table **324**, access frequency table **325**, link frequency table **326**, external access frequency table **327**, and external link frequency table **328**.

The home page cache **104** is the main cache, and stores therein generally a plurality of pairs of a URL **401** and URL contents **402**.

The access strategy table **105** is a combination of the cache directory **323** and communications cost table **324** to be described below. More specifically, the access strategy table **105** stores therein generally a plurality of combinations of a URL **411**, a host **412**, a communications cost **413**, a URL presence probability **414**, and a size **415**. The URL **411** is a URL itself. The host **412** is a name of the server **205** or a name of the host having an information source. The communication cost **413** is represented by using communications throughput and latency. The URL presence probability **414** is a probability of a presence of a home page designated by URL **411** in the computer designated by the host **412**. The size **415** indicates the number of bytes of URL **411** at the previous last access. The access strategy table **105** is used for determining one or more second servers **205'**, **205"**, . . . to time **432** during a predetermined past period. The communications number **434** is the number of communications with the host at the time **432** during the predetermined past period.

The access frequency table **325** stores therein generally a plurality of combinations of a URL **441**, an access frequency **442**, and a size **443**. The URL **441** is a URL itself. The access frequency **442** is the number of accesses by the server **205** to a home page represented by URL **441** during a predetermined past period C1. The access frequency **442** of this embodiment is a counter group for counting the number of accesses at what time in what day of the week during the past week. The size **443** indicates the number of bytes of URL **441** at the preceding access.

The link frequency table **326** stores therein generally a plurality of combinations of a referencing URL **451**, a referenced URL **452**, and an access frequency **453**. The access frequency **453** indicates the number of accesses by the server **205** to a link from the referencing URL **451** to the referenced URL **452** during a predetermine past period C2. The access frequency **453** of this embodiment is a counter group for counting the number of accesses at what time in what day of the week during the past week. The external access frequency table **327** stores therein generally a plurality of combinations of a URL **461** and an access frequency **462**. The URL **461** is a URL itself. The access frequency **462** indicates the number of accesses by the other servers **205'**, **205"**, . . . of the present system to a home page designated by URL **461** during the predetermined past period C1. The access frequency **462** of this embodiment is a counter group for counting the number of accesses at what time in what day of the week during the past week.

The external link frequency table **328** stores therein generally a plurality of combinations of a referencing URL **471**, a referenced URL **472**, and an access frequency **473**. The access frequency **473** indicates the number of accesses by the other servers **205'**, **205"**, . . . to a link from the referencing URL **471** to the referenced URL **472** during the predetermined past period C2. The access frequency **472** of this embodiment is a counter group for counting the number of accesses at what time in what day of the week during the past week.

The host table stores combinations of host names. The host names in the host table include those of all the servers **205**, **205'**, **205"**, . . . on the computers **203**, **203'**, **203"**, . . . participating the present system.

The server **205** uses variables such as a prefetch flag, a new URL flag, and a previous last accessed URL. The prefetch flag is a variable of truth/false indicating whether the server **205** is presently prefetching or processing a request from the client **204** (or other servers **205'**, **205"**, . . .). The initial value is false. The new URL flag is a variable of truth/false indicating whether the received URL contents are newly added to the home page cache **104**. The initial value is false. The previous last accessed URL is a URL returned in response to the previous last client request. The initial value is "empty".

FIG. **5** shows the structures of the home page group table **107**, host URL message **124**, and host frequency message **125**.

The home page group table **107** stores therein generally a plurality of combinations of a start URL **481** and a URL group **482**. Each combination is a collection of URLs often accessed consecutively. The start URL **481** is single, and the URL group **482** includes one or more URLs, URL **483**, URL **483'**, URL483", . . .

The host URL message **124** is data containing a combination of a host **501**, a URL **502** and a flag **503**. One host URL message **124** may contain two or more combinations. The host **501** indicates a host name, the URL **502** is a URL itself, and the flag **503** is a variable of truth/false. The host URL message **124** is used for notifying another server of whether the server designated by the host **501** has a URL designated by URL **502** (flag **503** is truth) or not (flag **503** is false).

The host frequency message **125** is data containing a combination of a host **511**, a referencing URL **512**, a referenced URL **513**, and a frequency **514**. The host **511** indicates a host name, the referencing URL **512** is a URL itself, the referenced URL **513** is a URL itself or "empty", and the frequency **514** is a counter. The host frequency message **125** is used for notifying another server **205'**, **205"**, . . . of the frequency **514**, or the number of accesses by the server **205** designated by the host **511** to the link from the URL designated by the referencing URL **512** or the referencing URL to the referenced URL **513** during a predetermined past period. The values C1 and C2 may be changed when the server **205** is compiled or started up or

while it is operated. The access frequencies **442, 453, 462, 473,** and **514** store the number of accesses at what time in what day of the week during the past week. These frequencies are illustrative only and are not intended to limit the scope of the claim. The objective of the access frequency table **325**, link frequency table **326**, external access frequency table **327** and external link frequency table **328** is to record the number of accesses to what URL or what link. Other information may be recorded so long as the objective is satisfied. For example, in combination with or in place of the access frequencies **442, 453, 462, 473,** and **514**, final access times may be recorded to estimate the number of accesses to what URL or what link by using a least recently used (LRU) method.

With reference to FIGS. **1, 3, 6** and **7**, the process flow of the first server **205** will be described assuming that the first server **205** is requested from the client **204** or other servers **205', 205", . . .** to acquire the URL contents. The request input unit **301** of the first server **205** waits for a request from the client **204** or other servers **205', 205", . . .**, and when there is a request, it is received **(601, 350)**. The URL contents to be acquired are represented by a URL in the request from the client **204** or other servers **205', 205", . . .** A URL contained in the request from the client **204** or other servers **205', 205", . . .** will be called hereinafter a "subject URL". In the first server **205**, the subject URL is passed to the cache table referring unit **302 (351)**. In this case, the prefetch flag is set with a "false".

The cache table referring unit **302** searches a combination having the URL **401** same as the subject URL from the home page cache **104 (602, 380)**. If such a combination is present **(603)**, a control is passed to the response unit **305 (363)**, whereas if such a combination is not present **(604)**, a process is passed to the process strategy unit **303 (352)**.

With reference to the access strategy table **105**, the process strategy unit **303** selects one or more second servers **205', 205", . . .** and information sources to which the request is transferred **(381)**. First, a combination having URL **411** same as the subject URL is searched from the access strategy table **105 (605)** to check whether there is one or more such combinations **(606)**. If present **(607)**, one or more combinations are scheduled and selected by a method to be described later **(608)**. For each of one or more combinations selected by the scheduling and selection, the request is transferred to the servers or information sources designated by the host **412** in the order from a higher priority level scheduled **(610, 365)**. If there is no such combination **(609)**, the request is transferred to the information source of the subject URL **(610, 365)**. In both of the above cases, after the request is transferred, a process is passed to the receiver unit **304 (353)**. The servers **205', 205", . . .** and information sources to which the request was transferred are hereinafter called "request transfer destinations".

For the scheduling and selection, various algorithms may be used. Important indices are latency, throughput and communications size, from the viewpoint of network communications efficiency. Also important is a probability of a presence of information corresponding to the subject URL at the request transfer destination. In this embodiment, the following specific scheduling and selection are performed. Although the following D1, D2, D3, D4, D5 and D6 are constants, they may be changed when the server **205** is compiled or started up or while it is operated. The scheduling may be used in combination with other information possessed by the server **205**, without any practical problems. For example, as the other information, information provided by a system manager to the server **205**, information of similarity of an IP address, information of routing topology and the number of hops, information of a physical structure of communications lines, and the like may be used.

The sequential processes of the scheduling and selection are as follows. Of the combinations in the access strategy table **105** having URL **411** same as the subject URL, those having a throughput constant of D1 or lower in the communication cost **413** are excluded, those having a latency constant of D2 or lower in the communication cost **413** are excluded, and those having a constant of D3 or lower of the URL presence probability **414** in the communication cost **413** are excluded. A priority level (the larger the level, the higher the priority) is calculated by using an equation [URL presence probability **414**/((latency of the communication cost **413**+size **415**)/(through put of the communication cost **413**))], and combinations having a constant D4 or higher are selected. In this case, if a computer of the information source contained in the subject URL is not selected, this information source is selected as having the smallest priority level. The above sequential processes are the scheduling and selection of this embodiment.

If the request is transferred to a plurality of hosts **412** at the same time, responses are returned from a plurality of request transfer destinations nearly at the same time and the network near the first server **205** may become congested. On the other hand, if the request is transferred sequentially to the plurality of hosts **412** and each response is awaited each time the request is transferred, the resultant time taken to acquire the URL contents may be delayed. In order to alleviate these problems, the following communications protocol may by used. This protocol will be described with reference to FIGS. **8A** and **8B** in which communications between the server **205** and the request transfer destinations is illustrated in the time axis from the earlier time to the later time. FIGS. **8A** and **8B** illustrate operations under two different conditions.

As shown in FIG. **8A**, the request is transferred without being changed to the D5 request transfer destinations having higher priority levels (hereinafter called priority request transfer destinations **801, 801', . . .**) **(803)**. Upon reception of the request, each of the priority request transfer destinations immediately sends the URL contents corresponding to the subject URL back to the first server **205** if the destination has the URL contents **(804)**. Transferring the request without changing it means a request to immediately send the URL contents if the priority request transfer destination has the URL contents.

Each of the priority request transfer destinations immediately sends the URL contents corresponding to the subject URL back to the first server **205** if the destination has the URL contents, whereas if it has not the URL contents, it returns a message "there is no subject URL" **(805)**.

The first server **205** transfers a "response standby request" of the subject URL to request transfer destinations other than the priority request transfer destinations (hereinafter called non-priority request transfer destinations **802, 802', . . .**) **(806)**. The response standby request means to start responding if the subject URL is present and a response start request to be later described is received. If the non-priority request transfer destination has the URL contents corresponding to the subject URL, it stands by, whereas if not, it returns a message "there is no subject URL".

As shown in FIG. **8B**, if all the priority request transfer destinations send the message "there is no subject URL" **(805, 807)**, the first server **205** transfers a "response start request" **(808)** to one or more non-priority request transfer destinations which do not send the message "there is no

subject URL" as yet. Upon reception of the response start request, the non-priority request transfer destinations start sending the URL contents corresponding to the subject URL to the first server **205** if the destinations have the URL contents (**809**). Each of the non-priority request transfer destinations in a standby state releases this state if the response start request does not reach until a lapse of a predetermined time D6 after the reception of the response standby request.

This protocol alleviates congestion of the network near at the first server **205**, which congestion may occur if one or more priority request transfer destinations have the URL contents corresponding to the subject URL. This is because the request transfer destinations are classified into two groups and the maximum number of responses returned at the same time can be reduced. Furthermore, acquiring the URL contents corresponding to the subject URL is prevented from being delayed even if none of the priority request transfer destinations have the URL contents corresponding to the subject URL. This is because the first server **205** can transfer the response start request to the non-priority request transfer destinations without waiting for the response to the response standby request.

With the procedure of the scheduling and selection described above, since either the priority request transfer destinations or the non-priority request transfer destinations have the information source of the subject URL, it is to be noted that it is very rare that none of the request transfer destinations send the URL contents corresponding to the subject URL. However, if by any chance the URL contents corresponding to the subject URL are not sent back, receiving the URL contents corresponding to the subject URL is abandoned, in this embodiment. As a modification of this embodiment, a request may be transferred to other selected request transfer destinations. As a further modification, receiving the URL contents corresponding to the subject URL may be tried again after a lapse of a predetermined time.

The server **205** of this embodiment communicates with its clients **204**, **204'**, **204"**, . . . provided with services of the server **205**, as well as with other servers. The communications with other servers are essential for using the performance of the cache of the server **205** as much as possible. However, there is a fear that the communications with other servers may lower the essential services to the clients **204**, **204'**, **204"**, . . . In order to solve this problem, the server **205** suppresses requests from other servers to a predetermined amount per unit time. In this embodiment, the server **205** does not accept requests from other servers during a T1 period if the URL contents more than L1 bytes were transferred to other servers during the past T1 period. In another embodiment of the invention, the server **205** does not accept requests from other servers during the T1 period if the URL contents were transferred to other servers more than L2 times during the past T1 period. The constants T1, L1 and L2 may be changed when the server **205** is compiled or started up or while it is operated.

The receiver unit **304** waits for the URL contents corresponding to the subject URL to be sent back from one or more request transfer destinations (**366**), and thereafter the communications cost table **324** is updated.

A response for the subject URL is waited for during a predetermined period T2 (**611**). It is checked whether there is a response (**612**). If there is a response (**613**), the URL contents contained in the response are stored as a new combination in the home page cache **104** (**614**, **382**). In this case, the subject URL is used as a URL **401** of the new combination, and the received URL contents are used as URL contents **402**. It is checked whether the total memory amount of the home page cache **104** exceeds a constant C3 after the reception of the URL contents corresponding to the subject URL (or during the reception thereof) (**615**). A control is passed to the cache replacing unit **310** to be later described to perform a cache replacing process, only if the amount exceeds the constant C3 (**616**, **617** in FIG. **6**, and **355**, **356** in FIG. **3**). If the URL contents corresponding to the subject URL are not received (**619**), i.e., if all the request transfer destinations send the message "there is no subject URL", then a control is passed to the response unit **305** without processing the cache **104** (**354**). The constants T2 and T3 may be changed when the server **205** is compiled or started up or while it is operated.

Updating the communications cost table **324** is performed by the procedure illustrated in FIG. **9**. When the URL contents corresponding to the subject URL start being received from a first request transfer destination (**910**), the combination in the communications table **324** corresponding to the first request transfer destination is updated. This updating is performed by the following procedure.

(1) A combination is searched, having the host **431** same as the first request transfer destination and the time **432** same as the current time (**911**). The latency of the communications cost **433** is calculated again and the communications number **434** is incremented by 1 (**912**). If the combination cannot be found at **911**, a new combination is formed in the communications cost table **324** and initialized as having the host **431** same as the first request transfer destination, the time **432** same as the current time, and the communications number **434** of "1". The latency of the communications cost **433** is initialized to the current latency (a time from a request transfer to the current time if the first request transfer destination is the priority request transfer destination, or a time from a response start request transfer to the current time if the first request transfer destination is the non-priority request transfer destination). If the combination is found at **911**, the latency of the communications cost **433** is calculated again by using the current latency and the values of the communications cost **433** and communications number **434**, and the communications number **434** is incremented by 1.

(2) Next, a reception end time ENDTIME of the first request transfer destination is estimated through calculation (**913**). This calculation is performed in the following manner. A combination is searched from the access strategy table **105**, having URL **411** same as the subject URL and the host **412** same as the first request transfer destination. By using the values of the communications cost **413** and size **415** of the searched combination, ENDTIME is calculated as (current time+(size **415**/throughput of communications cost **413**)).

(3) Next, either a start of the URL contents corresponding to the subject URL from a second request transfer destination or an end of reception from the first request transfer destination is waited (**914**).

(4) It is checked whether the reception from the first request transfer destination has been completed (**915**). If a reception from the second request transfer destination starts before the reception from the first request transfer destination is completed (**916**), a combination in the communications cost table **324** corresponding to the second request transfer destination is updated in the similar manner to the first request transfer destination (**917**), and a reception end time ENDTIME2 from the second request transfer destination is calculated in the similar manner to the reception end time ENDTIME of the first request transfer destination

(918). It is checked whether ENDTIME is larger than ENDTIME2+a constant C4 (919). If larger (9920), a reception of the first request transfer destination is stopped (921), the first request transfer destination is replaced by the second request transfer destination (922), and ENDTIME2 is set to ENDTIME (923) to return to 914. The constant C4 may be changed when the server 205 is compiled or started up or while it is operated. If ENDTIME is equal to or lower than ENDTIME2+the constant C4 (924), a reception from the second request transfer destination is stopped (925).

If a reception from the first request transfer destination is completed at 915 (926), the throughput of the communications cost 433 of a combination corresponding to the first request transfer destination in the communications cost table 324 is calculated from a current throughput and the values of the communications cost 433 and communications number 434 (927). The current throughput is obtained by dividing the size of the received URL contents by a time from the request transfer to the current time if the first request transfer destination is the priority request transfer destination, or by a time from the response start request transfer to the current time if the first request transfer destination is the non-priority request transfer destination. The new URL flag is set with truth (928). The updating procedure of the communications cost table 324 has been described above.

The response unit 305 operates in the following manner. It is checked whether the prefetch flag is "truth" (620). If truth, a control is passed to the prefetch strategy unit 308 (621, 364). If the prefetch flag is "false" (622), it is checked whether the URL contents corresponding to the subject URL have been received (623). If not received (624), the message "there is no subject URL" is sent to the client 204 or another server received the request (367), and thereafter (625) a control is passed to the request input unit 301 to wait for a request from a new client 204 or another server (626). If the URL contents corresponding to the subject URL have been received (627), the URL contents are sent to the client 204 or another server received the request (628, 367).

It is checked whether the new URL flag is truth (529). If truth (630), after the host URL message 124 is transmitted (631), a control is passed to the link statistics updating unit 306 (632, 357). If the new URL flag is false (636), a control is passed directly to the link statistics updating unit 306 (357).

If the new URL flag is truth, it means that one or more combinations are added to the home page cache 104. Therefore, the host URL message 124 is transmitted. After the new URL flag is set with false, a new host URL message 124 having one combination is generated. The host name of the first server 205 is set to the host 601 of the combination, the subject URL is set to URL 502, and truth is set to the flag 503. The host URL message 124 having the combination is transmitted to one or more other servers 205', 205", . . . contained in the host table. For example, of the combinations in the host table, a combination containing the first server 205 is searched, and the host URL message 124 generated in the above manner may be transmitted to one or more host names excepting the host name of the first server 205. Alternatively, the host URL message 124 having the combination may be transmitted to each host name stored in the combinations in the host table. Instead of transmitting the host URL message 124 having one combination to the other servers 205', 205", . . . , the host URL message 124 having collected several combinations may be transmitted to the other servers 205', 205", . . . The first method can limit a communications coverage to relatively near servers 205', 205", . . . , and the last method can reduce the communications number.

The link statistics updating unit 306 updates the access frequency table 325 and link frequency table 326 to thereby record the access frequency of the subject URL and the access frequency of a specific link to the subject URL (386). This procedure will be detained hereinunder.

Of the combinations in the access frequency table 325, a combination having URL 441 same as the subject URL is searched (701). If such a combination is not present, a new combination is formed in the access frequency table 325. The new combination is initialized as having URL same as the subject URL, and all 0s of the counter group of the access frequency 442. A counter corresponding to the current time of the access frequency 442 in the searched or newly formed combination is incremented by 1, and the size 443 is stored with the number of bytes of the URL contents corresponding to the subject URL.

Next, it is checked whether there is a hyper text link to the subject URL from the previous last accessed URL (702). If such a link is present (703), the link frequency table 326 is updated (704). Specifically, in updating the link frequency table 326, a combination is searched from the combinations in the link frequency table 326, having the referencing URL 451 same as the previous last accessed URL and the referenced URL 452 same as the subject URL. If such a combination is not present, a new combination is formed in the link frequency table 326, and initialized as having the referencing URL 451 same as the previous last accessed URL, the referenced URL 452 same as the subject URL, and all 0s of the counter group of the access frequency 453. A counter corresponding to the current time in the access frequency 453 is incremented by 1. After the subject URL is set to the previous last accessed URL, a control is passed to the group updating unit 307 (705, 358).

If it is judged that there is not a link in step 702 (706), the subject URL is set to the previous last accessed URL and a control is passed to the prefetch statistics unit 308.

In the group updating unit 307, the home page group table 107 is updated (707, 387). Consider a collection of URLs frequently accessed consecutively, it can be thought of that there is a small number of URLs first accessed when this collection is accessed. In this embodiment, only when such a small number of URLs are found, a control is passed from the link statistics updating unit 306 to the group updating unit 307 under the above-described judgement of the link statistics updating unit 306.

The group updating unit 307 operates in the following manner. A combination is searched from the home page group table 107, having the start URL 481 same as the subject URL. If there is such a combination, this combination is deleted from the home page group table 107. Next, a new combination is formed in the home page group table 107 and initialized as having the start URL 481 same as the subject URL, and the URL group 482 of one or more URLs having a probability of a constant C5 or higher of being consecutively accessed from the subject URL. This probability of being consecutively accessed from the subject URL can be calculated by using the access frequency table 325, external access frequency table 327, link frequency table 326, and external link frequency table 328. The constant C5 may be changed when the server 205 is compiled or started up or while it is operated. The probability of consecutively accessing the second URL from the first URL can be estimated as (B/A)x(D/C), where: A is an access frequency of the first URL (represented by a total value of the counter group of the access frequency 442 in the combination in the access frequency table 325 having URL 441 same as the first URL); B is an access frequency of the

second URL from the first URL (represented by a total value of the counter group of the access frequency **453** in the combination in the link frequency table **326** having the referencing URL **451** same as the first URL and the referenced URL same as the second URL); C is an external access frequency of the first URL (represented by a value of the counter of the access frequency **462** in the combination in the external access frequency table **327** having URL **461** same as the first URL; and D is an external access frequency of the link from the first URL to the second URL (represented by a value of the counter of the access frequency **473** in the combination in the external link frequency table **328** having the referencing URL **471** same as the first URL and the referenced URL **472** same as the second URL). If the value A or B is 0, the probability of consecutively accessing the second URL from the first URL is 0. If the value C or D is 0, the probability of consecutively accessing the second URL from the first URL is set to (B/A).

There is a case wherein a link from the first URL to the second URL is not present, but there are a link from the first URL to a third URL, a link from the third URL to a fourth URL, . . . , a link from an N-th URL to the second URL. In this case, the probability of consecutively accessing the second URL from the first URL is p2xp3x, . . . , xpN, where: p2 is a probability of consecutively accessing the third URL from the first URL; p3 is a probability of consecutively accessing the fourth URL from the third URL, . . . , pN is a probability of consecutively accessing the second URL from the N-th URL. For such transition closure calculation, a well know method such as Dijkspra algorithm may be used which is described in a document "Data Structure and Algorithm" by Aho, Baifukan, Information Processing Series 11, 1987, at Section 6.3.

After the above processes, a control is passed to a prefetch statistics unit **308** (**359**).

The prefetch strategy unit **308** determines the URL contents corresponding to the prefetch URL group including one or more URLs likely to be accessed immediately after the subject URL is accessed, and receives the prefetch group URL group at the home page cache **104** without a request from the client **204** or other servers.

It is first checked whether a request has been received from the client **204** or other servers (**708**). If received (**709**), the prefetch flag is set with "false" and a control is passed to the request input unit **301**.

If a request has not been received (**710**), it is checked whether the prefetch flag is truth (**711**).

If the prefetch flag is false (**712**), a combination containing the subject URL is searched from the home page group table **107** (**388**). One or more URLs other than the subject URL contained in the searched combination is set to the prefetch URL group (**713**). At this time, the prefetch flag is set with "truth".

If the prefetch flag is "truth" (**714**), URL under prefetch is delated from the prefetch URL group (**715**). In this case, if the prefetch URL group becomes empty, the prefetch flag is set with "false".

It is checked whether one or more URLs are contained in the prefetch group (**718**). If contained (**717**), URL under prefetch is picked up from the prefetch URL group (**718**) and is considered as the subject URL to thereafter pass a control to the cache table referring unit (**361**) and start receiving URL under prefetch in the manner described earlier. If not contained (**718**), a control is passed to the prefetch scheduling unit **309** (**360**). As apparent from the above procedure, prefetch is activated by a request not only from the client but also from other servers. Namely, a plurality of servers distributed hierarchically can perform a series of prefetch.

The prefetch scheduling unit **309** prepares for acquiring URL when the network becomes less congested, URL being considered to be accessed in the future and not present in the home page cache **104**. The prefetch scheduling unit **309** operates in the following manner.

A combination having the start URL **481** same as the previous last accessed URL is searched from the home page group table **107** to select the combination corresponding to the previous last accessed URL from the home page group table **107**.

If such a combination is present, a prefetch time for each of one or more first URLs stored in the URL group **482** is determined from the communications cost table **324** and cache directory **323** (**721**, **384**, **385**). This is performed as in the following.

If the URL contents of the first URL are present in the home page cache **104**, no process is executed, whereas if not present, a combination having URL **421** same as the first URL is searched from the cache directory **323**. If such a combination is not present, no process is executed, whereas if there are one or more combinations, one combination is selected which has the highest URL presence probability **423**, and the host **422** of this selected combination is used as the host to be accessed in the future. Next, a first combination is searched from the communications cost table **324**, having the host **431** same as the host of the selected combination. If such a combination is not present, no process is executed. If there are a plurality of first combinations, a second combination is searched from the access frequency table **325**, having URL **441** same as the first URL. If there is such a second combination, a third combination is selected from the first combinations, which has a minimum value of ((size **443** of the second combination/throughput of the communications cost **433** of the first combination)+latency of the communications cost **433** of the first combination). If there is no second combination, one third combination is selected which has a minimum value of (latency of the communications cost **433** of the first combination/throughput of the communications cost **433** of the first combination).

A time next corresponding to the time **432** of the third combination is called a "prefetch time". For example, if the time **432** is "Wednesday, 23:40" and the current time is January 3, Tuesday, 3:00", then the prefetch time is January 4, Wednesday, 23:40.

Next, the prefetch time of one or more URLs calculated at **721** is set to the timer **311** (**722**, **389**). The timer **311** operates to pass the request of the first URL to the cache table referring unit **302** at the time **432** in order to receive the URL contents corresponding to the first URL in the manner described earlier. After this setting process of the timer **311**, a control is passed to the request input unit **301** (**362**).

The procedure of the prefetch scheduling unit **309** has been described above.

<Cache Replacement>

The cache replacing unit **310** partially deletes the home page cache **104** (**383**). The following "cache value" of each of some or all combinations in the home page cache **104** is calculated.

The cache value of the first URL is calculated by using the access strategy table **105** and access frequency table **325**. One or more first combinations are searched from the access strategy table **105**, having URL **411** same as the first URL. If there are a plurality of first combinations, one first combination having a minimum value of (latency of the communications cost **413**+(size **415**/throughput of the communications cost **413**)) is selected. A second combination is

searched from the access frequency table **325**, having URL **411** same as the first URL.

The cache value of the first URL is ((AxB)/size **415** of the first combination), where A is ((size **41**/throughput of the communications cost **413**)+latency of the communications cost **413**) of the first combination, and B is the access frequency of the first URL (i.e., a total value of the counter group of the access frequency **442** of the second combination). If by any change there is no first or second combination, the cache value of the first URL is considered to be 0.

After the cache value of each of some or all combinations in the home page cache **104** is calculated, combinations in the home page cache **104** are deleted in the order from the minimum cache value. This deletion from the home page cache **104** continues until the total of combinations in the home page cache **104** becomes equal to or smaller than the constant C3.

If one or more combinations are deleted from the home page cache **104**, the host URL message **124** is transmitted by the following procedure. The host URL message **124** is newly generated, and a new combination is formed in the host URL message **124** for each of the deleted combinations. Each new combination has the host **501** same as the host name of the first server **205**, URL **502** same as URL **401** of the deleted combination, and flag **503** of false. The generated host URL message **124** is transmitted to one or more other servers **205'**, **205"**, . . . contained in the host table. A method of determining the transmission destinations is the same as the case of the host URL message **124** which is used when one or more combinations are added to the home page cache **104**.

<Timer Process>

The server **205** performs the above-described series of processes as well as a process which is periodically performed in response to the timer **311**. Each of the following processes to be executed in response to the timer is performed only while the request input unit **301** waits for a request from the client **204** or other servers, in order to avoid any conflict with the main flow.

The timer **311** operates to periodically reconfigure the access strategy table **105** by using the cache directory **323**, communications cost table **324** and access frequency table **325**. The reason for this is that although the communications cost table **324** mainly stores the communications costs with each host at what time in what day of the week at the time **432**, the access strategy table **105** stores current communications costs with each host. In this embodiment, since the time **432** stores information of each hour, the access frequency table **325** is reconfigured once per hour. The reconfiguring procedure is apparent from the contents of the access strategy table **105**, cache directory **323**, communications cost table **324** and access frequency table **325**.

The timer **311** operates to periodically generate the host frequency message **125** from the access frequency table **325** and link frequency table **326** and transmit it to one or more other servers **205'**, **205"**, . . . contained in the host table. This procedure is performed in the following manner.

First, one host frequency message **125** is generated. Next, a new combination is added to the host frequency message **125** for each of all combinations in the access frequency table **325**. The added combination has the host **511** same as the host name of the first server **205**, the referencing URL **512** same as URL **441** of the combination in the access frequency table **325**, the referenced URL **513** of "empty", and the frequency **514** of a total value of the counter group of the access frequency **442** of the combination in the access frequency table **325**. Next, a new combination is added to the host frequency message **125** for each of all combinations in the link frequency table **326**. The added combination has the host **511** same as the host name of the first server **205**, the referencing URL **512** same as the referencing URL **451** of the combination in the link frequency table **326**, the referenced URL **513** same as the referenced URL **452** of the combination in the link frequency table **325**, and the frequency **514** of a total value of the counter group of the access frequency **453** of the combination in the link frequency table **326**. Next, a combination having the first server **205** is searched from the combinations in the host table. In accordance with searched combinations, the host frequency message **125** generated by the above procedure is transmitted to one or more hosts excepting the host of the first sever **205**. The timer **311** also operates to transmit the host frequency message **125** generated by the above procedure to one or more hosts stored in combinations not containing the first server **205** searched from the host table at a frequency smaller than the transmission of the host frequency message **125** to the hosts stored in combinations containing the first server **205** (e.g., once per ten transmissions relative to the combinations containing the first server **205**).

The timer **311** also operates to periodically recalculate the URL presence probability **423** for each of all the combinations in the cache directory **323**. This is performed in the following manner. For each of the combinations in the cache directory **323**, the URL presence probability **423** subtracted by a constant C6 is stored in the URL presence probability **423**. If the subtracted result is equal to a constant C7 or smaller, C7 is stored in the URL presence probability **423**. In this case, the constants C6 and C7 may be changed when the server **205** is compiled or started up or while it is operated.

<Message Process>

When the inter-server message processing unit **103** of the first server **205** receives the host URL message **124** from other servers **205'**, **205"**, . . . , the cache directory **323** is updated by the following procedure. For each of first combinations with the flag **503** of truth in the host URL message **124**, a second combination having the host **422** same as the host **501** of the first combination and URL **421** same as URL **502** of the first combination is searched from combinations in the cache directory **323**. If there is such a second combination, the URL presence probability **423** of the second combination is set to 1. If there is no second combination, a new combination is formed in the cache directory **323** and initialized as having the host **422** same as the host **501**, URL **421** same as URL **502**, and the URL presence probability **423** of 1.

For each of third combinations with the flag **503** of false in the host URL message **124**, a fourth combination having the host **422** same as the host **501** of the third combination and URL **421** same as URL **502** of the third combination is searched from combinations in the cache directory **323**. If there is such a fourth combination, it is deleted.

When the inter-server message processing unit **103** of the first server **205** receives the host frequency message **125** from other servers **205'**, **205"**, . . . , the external access frequency table **327** and external link frequency table **328** are updated by the following procedure.

For each of first combinations with the referenced URL **513** of "empty" in the received host frequency message **125**, a second combination having URL **451** same as the referencing URL **512** of the first combination is searched from combinations in the external access frequency table **327**. If there is such a second combination, the frequency **514** of the

first combination is added to the access frequency **462** of the second combination, and the addition result is stored in the access frequency **462** of the second combination. If there is no second combination, a new combination is formed in the external access frequency table **327** and initialized as having URL **461** same as the referencing URL **512** of the first combination and the access frequency **462** same as the frequency **514** of the first combination.

For each of third combinations with the referenced URL **513** of "empty" in the host frequency message **125**, a fourth combination having referencing URL **471** same as the referencing URL **512** of the third combination and the referenced URL **472** same as the referenced URL **513** of the third combination is searched from combinations in the external link frequency table **328**. If there is such a fourth combination, the frequency **514** of the third combination is added to the access frequency **473** of the fourth combination, and the addition result is stored in the access frequency **473** of the fourth combination. If there is no fourth combination, a new combination is formed in the external link frequency table **328** and initialized as having the referencing URL **471** same as the referencing URL **512** of the third combination, the referenced URL **472** same as the referenced URL **513** of the third combination, and the access frequency **473** same as the frequency **514** of the third combination.

If the program realizing the invention method and to be executed by a computer is stored in a recording **58** medium, the invention method can be performed at a desired site.

According to the present invention, a user connects a desired server by using a client and requests information to the server. In this case, a plurality of servers can shield irregular and unstable natures of communications lines. Even if throughput and latency of each communications line between servers are different, each server exchanges communications information to select an optimum server which can acquire information at the highest speed. Therefore, irregular nature of communications lines can be eliminated by selecting and using a communications line which can acquire information at the highest speed. By exchanging communications information between servers, each server can take into consideration, for example, a change in line congestion with time zone and day of the week, and a change in a routing pattern to be caused by an enhancement of one communications line and hence new congestion or alleviated congestion of another communications line. Accordingly, each server selects another server which can acquire information at the highest speed and selects a communications line which can acquire information at the highest speed. In this manner, unstable nature of communications line can be avoided.

What is claimed is:

1. A distributed data management method used by a process as a client using one or more sets of data and by two or more processes as servers providing data designated by a request from the client, in a computer system having two or more computers each executing one or more processes and interconnected by a network, the method comprising the steps of:

(1) storing in memory of each of a plurality of servers past communications history between said each server and others of said servers;

(2) receiving from a first server a request for data, and selecting one or more third servers for transmitting the requested data at high speed from second servers which store the requested data based on communications history stored in the memory of said first server; and

(3) transmitting the requested data to said first server from said one or more third servers.

2. A distributed data management method according to claim **1**, wherein said storing step (1) includes a step of storing throughput representative of a transferrable data amount per unit time and latency representative of communications delay time.

3. A distributed data management method according to claim **1**, wherein said storing step (1) includes a step of storing the communications history including a size of the necessary data.

4. A distributed data management method according to claim **1**, wherein said storing step (1) includes a step of storing the communications history including a data storage probability predicted by said each server, said data storage probability indicating a possibility that said others of said servers store the requested data.

5. A distributed data management method according to claim **1**, wherein said selecting step (2) includes a step of predicting a time taken to complete the transfer of the necessary data from the communications history and a step of selecting one or more third servers from one or more second servers if the predicted communications completion time is equal to or shorter than a predetermined value.

6. A distributed data management method according to claim **5**, wherein the predicting step is performed by using an equation (the latency+(the size of the necessary data/the throughput)).

7. A distributed data management method according to claim **4**, wherein said selecting step (2) includes a step of selecting one or more third servers from one or more second servers if a product of the data storage probability and the communications completion time is equal to or smaller than a predetermined value.

8. A distributed data management method according to claim **4**, wherein said transmission requesting step (3) includes a step of updating the communications history of all of two or more servers stored in the memory, when communications between first and second servers is executed.

9. A distributed data management method used by two or more processes as servers in a computer system having two or more computers each executing one or more processes and interconnected by a network, comprising the steps of:

performing a process in a first server of requesting two or more second servers to transmit necessary data to said first server in accordance with a possibility of a presence of the necessary data in said two or more second servers,

wherein said process comprises the steps of:

(1) selecting by said first server one or more third servers from two or more second servers,

(2) requesting the selected third server to transmit the necessary data, and

(3) requesting the second server other than the third server to hold a transmission of the necessary data, and

wherein said selecting step (1) comprises the steps of:

storing by the first server past communications history between the first server and the second servers in a memory, and

selecting the third server from the second servers by using the past communications history.

10. A distributed data management method according to claim **9**, wherein said requesting step (2) includes a step of requesting by the first server some or all of the second servers not selected as the third server to immediately transmit the necessary data, if some or all of the third servers sends a message that the necessary data cannot be transmitted, to the first server.

11. A distributed data management method according to claim 10, wherein said requesting step (2) includes a step of continuing a transmission only from one or more fourth servers and stopping a transmission from other servers, after two or more second servers starts a transmission of the necessary data.

12. A distributed data management method according to claim 11, wherein said selecting step (1) includes a step of selecting the third server and the fourth server.

13. A distributed data management method used by a process as a client using one or more sets of data and by two or more processes as servers in a computer system having two or more computers each executing one or more processes and interconnected by a network, the method comprising the steps of:

(1) storing by a first server in a memory, past communications history with time during a first time at a second time interval between the first server and two or more second servers;

(2) predicting by the first server a time when prefetch data can be acquired at high speed from one or more second servers having a possibility of possessing the prefetch data, by using the communications history with time, in order to request servers other than the first server to acquire the prefetch data before a request from the client, the prefetch data being expected to have a high possibility to be requested by the client in a future; and

(3) requesting by the first server at the predicted time the third server selected from at least some of the second servers to transmit the prefetch data.

14. A distributed data management method according to claim 13, wherein said storing step (1) includes a step of storing the communications history with time containing a history of throughput of communications during the first time at the second time interval and a history of latency of communications during the first time at the second time interval.

15. A distributed data management method used by two or more processes as servers in a computer system having two or more computers each executing one or more processes and interconnected by a network, the method comprising the steps of:

(1) storing by a first server past communications history between said first server and second servers in a memory;

(2) selecting one or more third servers from second servers associated with the first server, by using the communications history; and

(3) transmitting from the first server to the second server at least part of a list of data possessed by the first server.

16. A distributed data management method according to claim 15, wherein said transmitting step (3) includes a step of determining a data presence probability of the first server in accordance with a difference between a time when the data list is transmitted and a current time.

17. A distributed data management method according to claim 16, wherein said transmitting step (3) includes a step of lowering the data presence probability at a predetermined time interval after the data presence probability is set to 1 when the data list is transmitted.

18. A distributed data management method used by a process as a client using one or more sets of data and by two or more processes as servers, in a computer system having two or more computers each executing one or more processes and interconnected by a network, the method comprising the steps of:

(1) storing at least one of past communications history between a first server and second servers and request history from the client to the first server in a memory,

(2) determining by the first server one or more second data sets having a higher frequency of a request following a request for a first data set in accordance with the request history;

(3) storing by the first server, as reference relationship information, data representative of a combination of a name of the first data set and names of second data sets; and

(4) exchanging the reference relationship information of the first server with reference relationship information of one or more second servers.

19. A distributed data management method according to claim 18, wherein said determining step (2) includes a step of determining by the first server one or more second data sets having a higher possibility of being requested after the first data set is requested by the client, in accordance with the reference relationship information and a step of prefetching the second data sets from one or more second servers having a possibility of possessing the second data sets.

20. A distributed data management method according to claim 19, wherein said prefetching step includes a step of selecting one or more second servers from one or more third servers associated with the first server, by using the communications history.

21. A distributed data management method used by a process as a client using one or more sets of data and by two or more processes as servers, in a computer system having two or more computers each executing one or more processes and interconnected by a network, the method comprising the steps of:

(1) receiving by a first server a transmission request of a first data set from a second server;

(2) selecting one or more second servers expected to store a second data set having a possibility of being requested after the first data set; and

(3) requesting the selected one or more second servers to transmit the second data set to hierarchically prefetch the second data set from the first server to one or more second servers, and

wherein said selecting step (2) comprises a step of:

selecting one or more third servers associated with the first server from one or more second servers by using past communications history between the first server and the second servers.

22. A distributed data management method used by a process as a client using one or more sets of data and by two or more processes as servers, in a computer system having two or more computers each executing one or more processes and interconnected by a network, the method comprising the steps of:

(1) predicting a time taken to acquire from the second server two or more data sets processed by the first server;

(2) scheduling an order of discarding two or more data sets by using the predicted time, and

wherein said predicting step (1) is executed by using past communications history between servers.

23. A distributed data management method according to claim 22, wherein said scheduling step (2) is executed by using the predicted time and the number of times the data set requested by the client in a predetermined past time period.

24. A computer program product comprising:

a computer useable medium having computer readable program code means embodied therein for performing distributed data management by a process as a client using one or more sets of data and by two or more processes as servers providing data designated by a

request from the client, in a computer system having two or more computers each executing one or more processes and interconnected by a network, the computer readable program code means in the computer program product comprising:

computer readable program code means for storing past communications history between first and second servers in a memory;

computer readable program code means for selecting one or more third servers from second servers by using the communications history, in accordance with a possibility that one or more second servers store necessary data for the first server; and

computer readable program code means for transmitting the necessary data from the first server to the third servers.

25. A distributed data management system comprising:

a network for interconnecting a computer as a client for using one or more data set and two or more computers as servers possessing one or more data sets for providing data designated by the client;

storage means for storing past communications history between a first server and one or more second servers;

means for selecting at least one second server having a possibility of possessing necessary data for the first server, in accordance with the past communications history; and

means for requesting the selected second server to transmit the necessary data.

26. A computer readable program product comprising:

a computer useable medium having a computer program embodied therein for performing distributed data management by two or more processes as servers in a computer system having two or more computers each executing one or more processes and interconnected by a network, said computer program comprising:

Computer program code means for requesting by a first server, two or more second servers to transmit necessary data to said first server in accordance with a possibility of a presence of the necessary data in said two or more second servers,

wherein said computer program code means comprises:

first computer program code means for selecting by said first server selecting one or more third servers from two or more second servers,

second computer program code means for requesting the selected third server to transmit the necessary data, and

third computer program code means for requesting the second server other than the third server to hold a transmission of the necessary data, and

wherein said first computer program code means comprises:

computer program code means for storing by the first server past communications history between the first server and the second servers in a memory, and

computer program code means for selecting the third server from the second servers by using the past communications history.

27. A computer readable program product comprising:

a computer useable medium having computer readable program code means embodied therein for performing distributed data management by a process as a client using one or more sets of data and by two or more processes as servers in a computer system having two or more computers each executing one or more processes and interconnected by a network, the computer readable program code means in the computer readable program product comprising:

computer readable program code means for storing by a first server in a memory, past communications history with time during a first time at a second time interval between the first server and two or more second servers;

computer readable program code means for predicting by the first server a time when prefetch data can be acquired at high speed from one or more second servers having a possibility of possessing the prefetch data, by using the communications history with time, in order to request servers other than the first server to acquire the prefetch data before a request from the client, the prefetch data being expected to have a high possibility to be requested by the client in a future; and

computer readable program code means for requesting by the first server at the predicted time the third server selected from at least some of the second servers to transmit the prefetch data.

28. A computer readable program product comprising:

a computer useable medium having computer readable program code means embodied therein for performing distributed data management by two or more processes as servers in a computer system having two or more computers each executing one or more processes and interconnected by a network, the computer readable program code means in the computer readable program product comprising:

computer readable program code means for storing by a first server past communications history between the first server and second servers in a memory;

computer readable program code means for selecting one or more third servers from second servers associated with the first server, by using the communications history; and

computer readable program code means for transmitting from the first server to the second server at least part of a list of data possessed by the first server.

29. A computer readable program product comprising:

a computer useable medium having computer readable program code means embodied therein for performing distributed data management by a process as a client using one or more sets of data and by two or more processes as servers, in a computer system having two or more computers each executing one or more processes and interconnected by a network, the computer readable program code means in the computer readable program product comprising:

computer readable program code means for storing at least one of past communications history between a first server and second servers and request history from the client to the first server in a memory,

computer readable program code means for determining by the first server one or more second data sets having a higher frequency of a request following a request for a first data set in accordance with the request history;

computer readable program code means for storing by the first server, as reference relationship information, data representative of a combination of a name of the first data set and names of second data sets; and

computer readable program code means for exchanging the reference relationship information of the first server with reference relationship information of one or more second servers.

* * * * *

APPENDIX D

U.S. Patent No. 5,867,667 to Butman *et al.*

US005867667A

United States Patent [19]

Butman et al.

[11] Patent Number: 5,867,667

[45] Date of Patent: Feb. 2, 1999

[54] **PUBLICATION NETWORK CONTROL SYSTEM USING DOMAIN AND CLIENT SIDE COMMUNICATIONS RESOURCE LOCATOR LISTS FOR MANAGING INFORMATION COMMUNICATIONS BETWEEN THE DOMAIN SERVER AND PUBLICATION SERVERS**

[75] Inventors: **Ronald A. Butman**, Nahant; **Raja Ramachandran**, Allston; **Thomas A. Burns**, Duxbury; **Thomas J. Malone**, South Boston; **Michael D. Kmiec**, Boston; **Joseph C. Dougherty**, West Roxbury, all of Mass.

[73] Assignee: **PFN, Inc.**, Cambridge, Mass.

[21] Appl. No.: **822,898**

[22] Filed: **Mar. 24, 1997**

[51] **Int. Cl.**[6] **G06F 13/00**

[52] **U.S. Cl.** **395/200.79**; 395/200.57; 395/200.55

[58] **Field of Search** 395/200.79, 200.57, 395/200.55, 200.59, 200.49, 200.53, 200.48

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,428,778	6/1995	Brookes	707/5
5,555,798	9/1996	Skeen et al.	705/35
5,572,643	11/1996	Judson	395/200.48
5,708,780	1/1998	Levergood et al.	395/200.59
5,734,831	3/1998	Sanders	395/200.53
5,764,906	6/1998	Edelstein et al.	395/200.49
5,774,670	6/1998	Montulli	395/200.57

OTHER PUBLICATIONS

CheckPoint Software Tech.Ltd., CheckPoint FireWall-1 White Paper—Sep. 1995—Published on Internet.

NCSA, NCSA Firewall Policy Guide—Feb. 1996—Published on Internet.

IBM, Intranet and client/server—Apr., 1996—Published on Internet.

David Strom, Creating Private Intranets: Challenges and Prospects for IS—Nov. 16, 1995—Publsihed on Internet.

CGI, A Sample forma nd its CGI script Common Gateway Interface.

D.R.T., The WWW Common Gateway Interface Version 1.1—Feb. 15, 1996—Published on Internet.

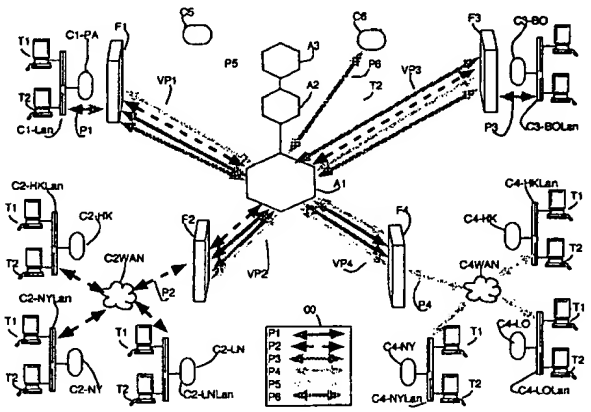
(List continued on next page.)

Primary Examiner—Le Hien Luu
Attorney, Agent, or Firm—Maureen Stretch

[57] **ABSTRACT**

A publication control system for networks inside a client having several publication computers networked together, each of the publication computers having disks for storing a dynamic group registry and resource locators containing function names, a web server which, when executed by the publication computer, causes the publication computer to respond to resource locators by calling the function indicated, a database management program for organizing the dynamic group registry; and a client side communications server, which responds to resource locators directed to it and directs the database management program in organizing the dynamic group registry; the system also has a domain computer having a disk for storing a dynamic client registry and resource locators containing function names; a web server which, when executed by the domain computer, causes the domain computer to respond to the resource locators by calling the function name indicated, a database management program for organizing the dynamic client registry; a domain communications server which, when loaded by the web server, is executed by the domain computer, to respond to resource locators directed to it and to direct the database management program in organizing the dynamic client registry; a domain communications resource locator list in all computers that causes functions to be executed in the domain communications server; a client side communications resource locator list in all computers that causes functions to be executed in each client side communications server so that communications between the domain computer and the publication computers cause the selected functions to control internal publications.

16 Claims, 45 Drawing Sheets



OTHER PUBLICATIONS

Russell Owen, ROFM, a Filemaker Pro CGI—Jul. 8, 1996—Published on Internet.
Andy Oram, Introducing "CGI Programming on the Web"—Published on Internet.
seriver, Server Push Animation—Feb. 17, 1996—Published on Internet.
William Graziadei, Decoding FORMS with CGI—1995—Published on Internet.
Internet Technologies, The Intranet–Revolution or Evolution ?—Apr. 21, 1996—Published on Internet.
Darrin Nelson, Lotus, Lotus Notes and the Internet Compared and Contrasted Draft—Dec. 1, 1995—Published on Internet.
Netscape Communications, Intranets Redefine Corporate Information Systems—1996—Published on Internet.
EXTRA CORP, —EXTRAnet—Oct. 23, 1996—Published on Internet.
John Wack, Packet Filtering Firewall—Feb. 9, 1995—Published on Internet.
TCP/IP, Networking for the Internet or an Intranet.
Borderware, The BorderWare Firewall Server 4.0 White Paper Version 1.0—Nov. 1996—Published on Internet.
iCat (Innergy Inc.), The Intranet FAQ—Oct. 31, 1996—Published on Internet.
MPI, Illustra Advantages for WWW Applications—Dec. 21, 1995—Published on Internet.
Web–Star, WebStar/SSL Security Toolkit—Published on Internet.
MacDNS, Macintosh Domain Name Server Frequently Asked Questions—Oct. 28, 1996—Published on Internet.
MacWeek Special Report, Macweek Guide to Intranets—Aug. 5, 1996, vol. 10, No. 30—Published on Internet.
Lee Levitt, Internet Technologies Deployed Behind the Firewall for Corporate Productivity Prepared for the Internet Society INET'96 Annual Meeting—Published on Internet.
Steven E. Newton, What is TCP/IP?—Jan. 20, 1994—Published on Internet.
Yahoo!, Intranet Bookmarks—1994–96—Published on Internet.
Netscap's Secure Sockets Layer (SSL).
Michael Sarkin, PC Week Labs Review, Introducing the Compaq Professional Workstation, "Security" Err on the side of caution when considering Internet connections—Oct. 30, 1995—Published on Internet.
Peter Hinxman, University of Wales, Getting the most out of TCP–wrapper—Nov. 1994—Published on Internet.
America Online, AOL Server: A Server Comparison; Examples; Categories—1996—Published on Internet.
network MCI WebMaker, NetworkMCI WebMaker Security Brief, Overview of the networkMCI Webmaker—Published on Internet.
Matt Kramer, PC week Tech View Lab, LDAP seeks to solve directory confusion—May 23, 1996—Published on Internet.
Cisco Systems Inc., Designing Large–Scale IP Internetworks—1998–1996—Published on Internet.
Steven Adler, Richard Sand, Internet Insurance: Property, Contents, and Commerce A White Paper, IBM—1996—Published on Internet.
Goscinny–Uderzo, Firewalls, adapted from the document "Internet Firewalls FAQ"—1991—Published on Internet.
Center for Technology in Government, Internet Security Seminar, Center for Technology in Government, Univeristy of Albany—Apr. 2, 1996–Published on Internet.
Ir. Rob Koreman, A Discussion of Security Rules for the use of Internet and the Web, a lecture within the 1995–1996 UIA Post–academic Program, Telecommunications and Telematica—Mar., 1996—Published on Internet.
Tina Darmohray, Marcus Ranum, Firewalls—1995—Published on Internet.
W. Yeong, T. Howes, S. Kille, Network Working Group Request for Comments: 1487—Jul., 1993.
Tobin Anthony, Building and Maintaining an Intranet with the Macintosh, pp. 304–309, Hayden Books.
Jerry Ablan, Scott Yanoff, Web Site Administrator's Survival Guide, pp. 221–226, Sams.Nets Publishing.
Lisa Pyle, Creating Lotus Notes Applications, Table 2.1, pp. 20–25—1994, Que Corporation.
Lisa Pyle, Creating Lotus Notes Applications, Our Case Study Continues: Extending the Automation Capabilities, pp. 274–294—1994, Que Corporation.
Lisa Pyle, Creating Lotus Notes Applications, Query Sharing, pp. 348–359—1994, Que Corporation.
Zahir Ebrahim, A Brief Tutorial on ATM—Mar. 5, 1992—Published on Internet.
Michael Robin, The Medium Is The Web—1996—Published on Internet.
Erica Roberts, Data Communications, Getting the Goods on SNA Gateways—May 1996—Published on Internet.
Peter Heywood, To Chee Eng, Data Communications, Global Supernets: Big Pipes, Big Promises . . . and One Big Problem—Sep. 21, 1995—Published on Internet.
Ernst & Young, Health Care Cybervision, Internet Primer—1996—Published on Internet.
David Willis, State of the WAN—Apr. 15, 1996—Published on Internet.
Stephanie Wilkinson, Boundless Bandwidth—May 14, 1996—PCWeek Online—Published on Internet.
ClarkNet Business Solutions, Point to ISDN service—Published on Internet.
Kelly Jackson Higgins, Intranet Virtual Realities—1996—Published on Internet.
Isis Distributed Systems, Ensuring Application Availability is a Mission Critical Function—Published on Internet.
William Robertson, IP Multicast and MBONE Services on the Berkeley Campus Network—Mar. 7, 1996—Published on Internet.
Tibco, Inc., MTP: Multicast Transport Protocol—1994–1996—Published on Internet.
Tibco, Inc., TIBCO Named by IDC as Largest Message–Oriented Middleware Vendor—Sep. 17, 1996—Published on Internet.
Marc Andreesen and the Netscape Product Team, The Netscape Intranet Vision and Product Roadmap—Version 1.0, revised Jun. 11, 1996—Published on Internet.
Netscape, Netscape and Oracle Sign Strategic Agreement to Integrate and distribute Flagship Products—Published on Internet.

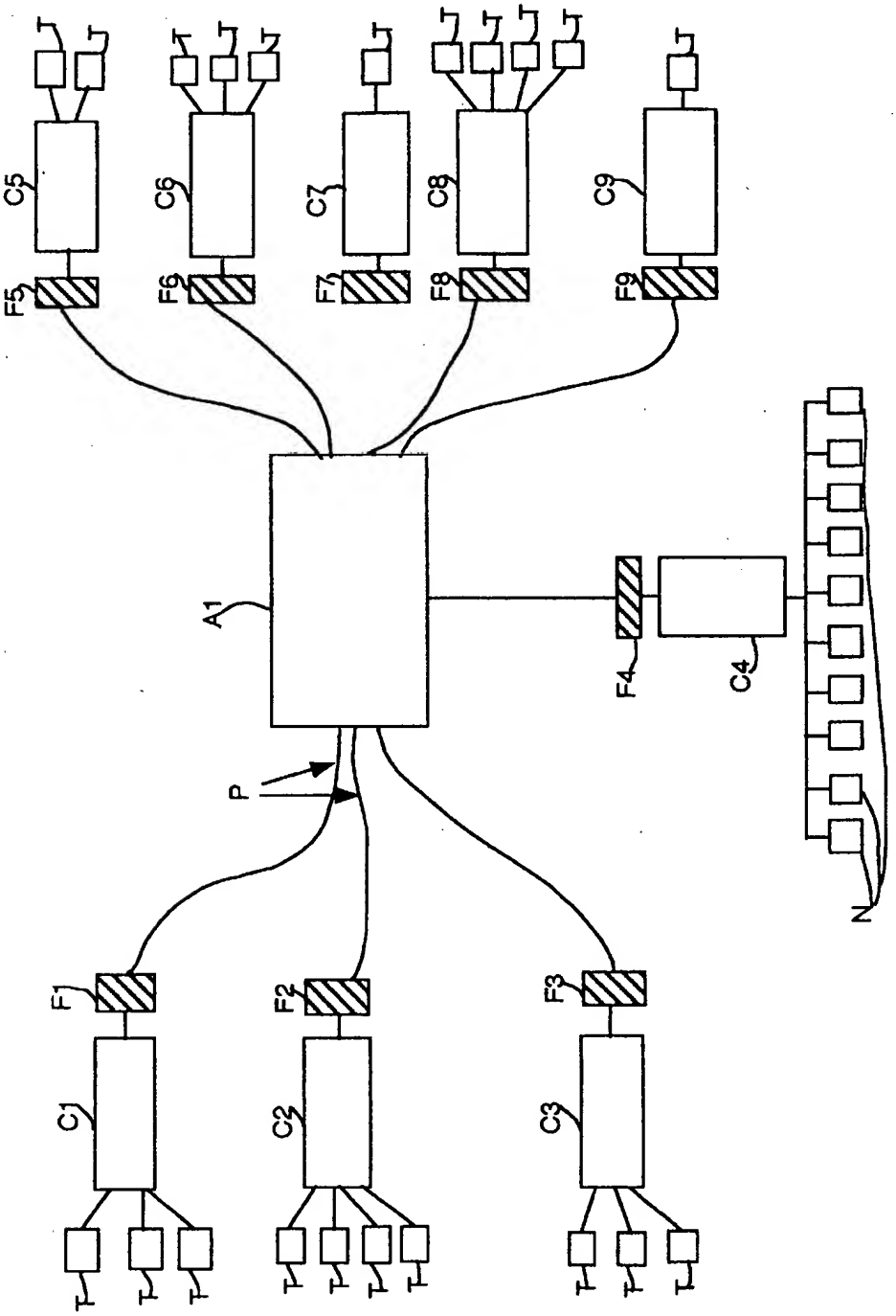


Fig. 1

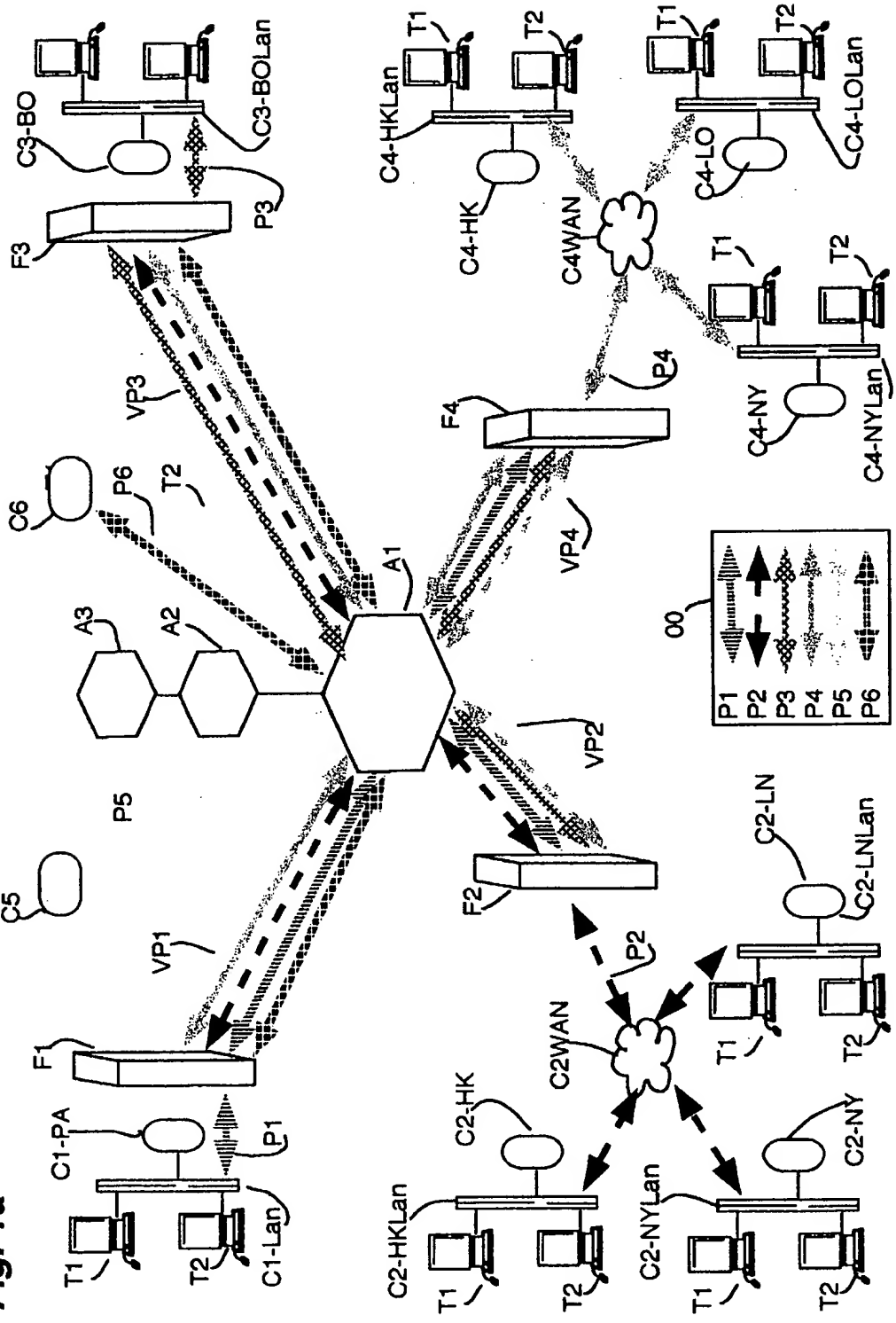

C3-BO
C3-BOLan
F3
P3
VP3
C6
P6
T2
A3
A2
A1
P5
C5
VP1
F1
C1-PA
P1
C1-Lan
T1
T2
C4-HKLan
C4-HK
C4WAN
C4-LO
C4-LOLan
C4-NY
C4-NYLan
F4
P4
VP4
VP2
F2
C2-LN
C2-LNLan
P2
C2WAN
C2-HK
C2-HKLan
C2-NY
C2-NYLan
00
P1
P2
P3
P4
P5
P6

Fig. 1a

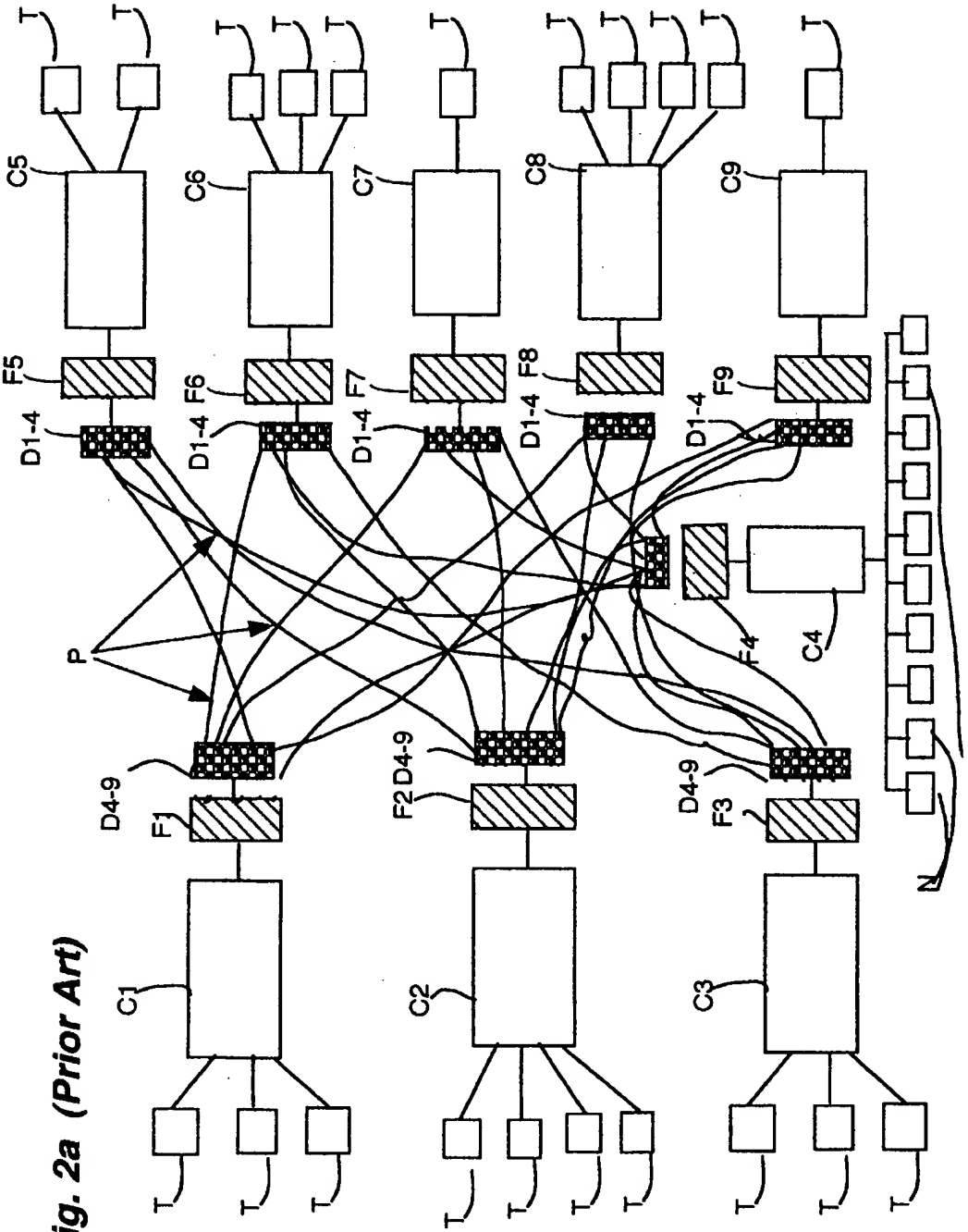


Fig. 2a (Prior Art)

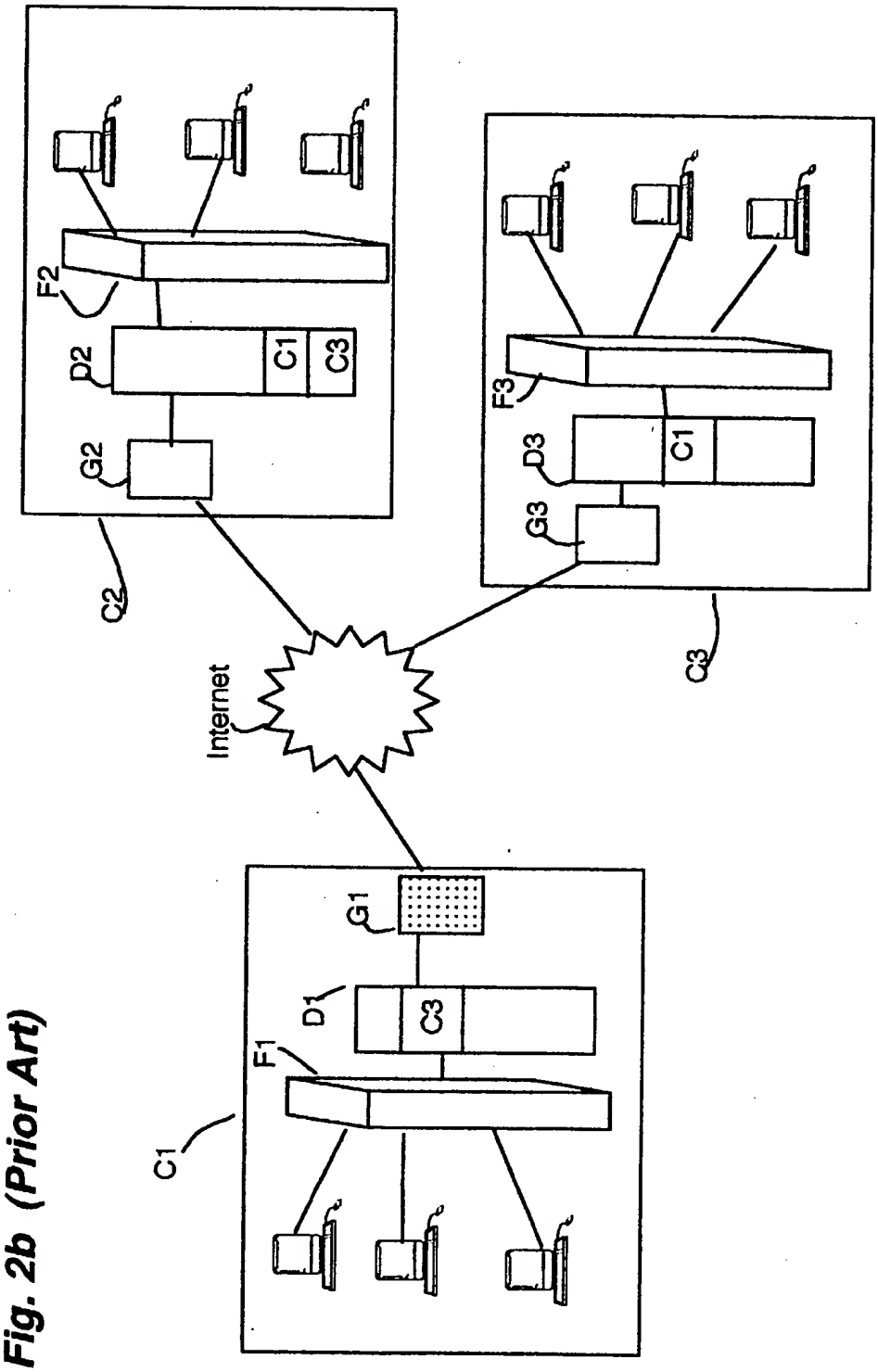


Fig. 2b (Prior Art)

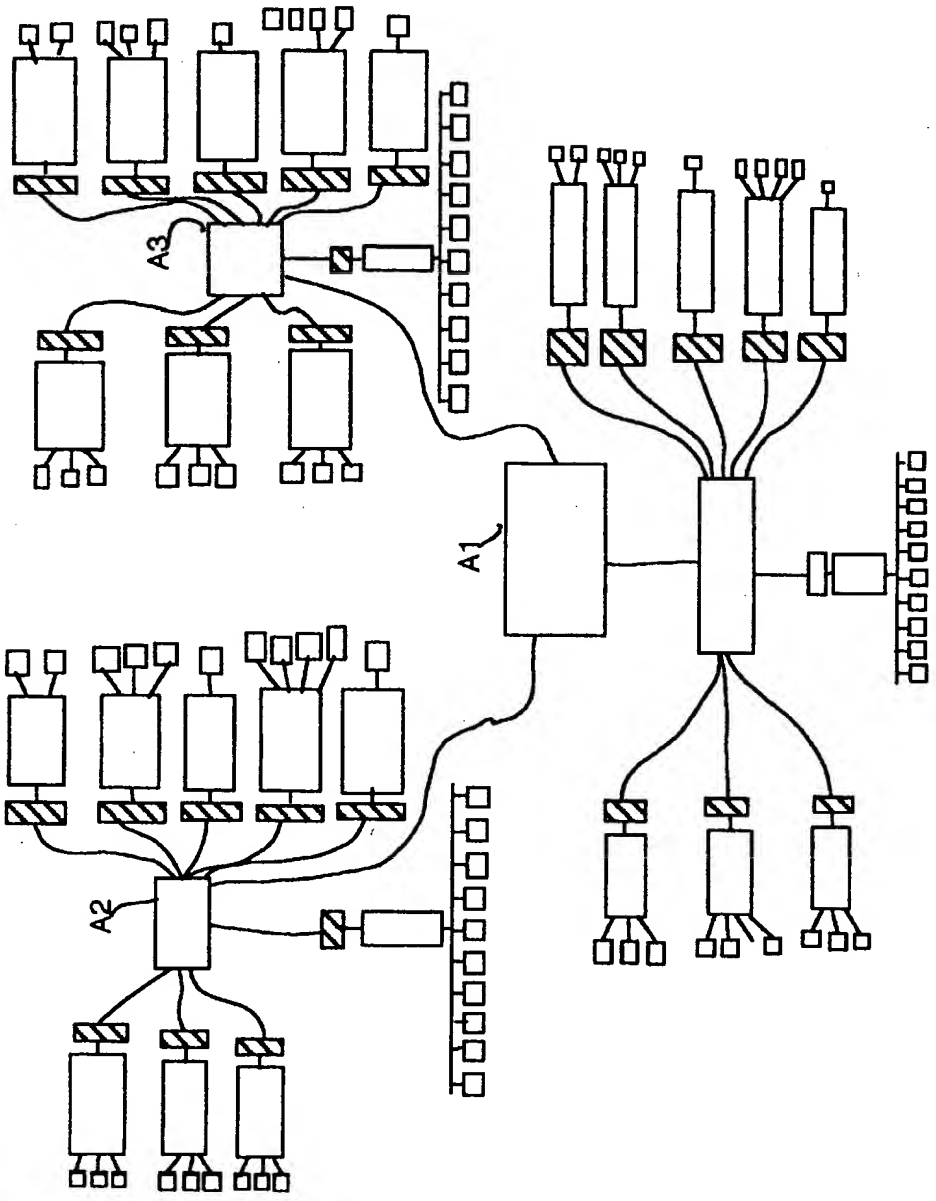


Fig. 3

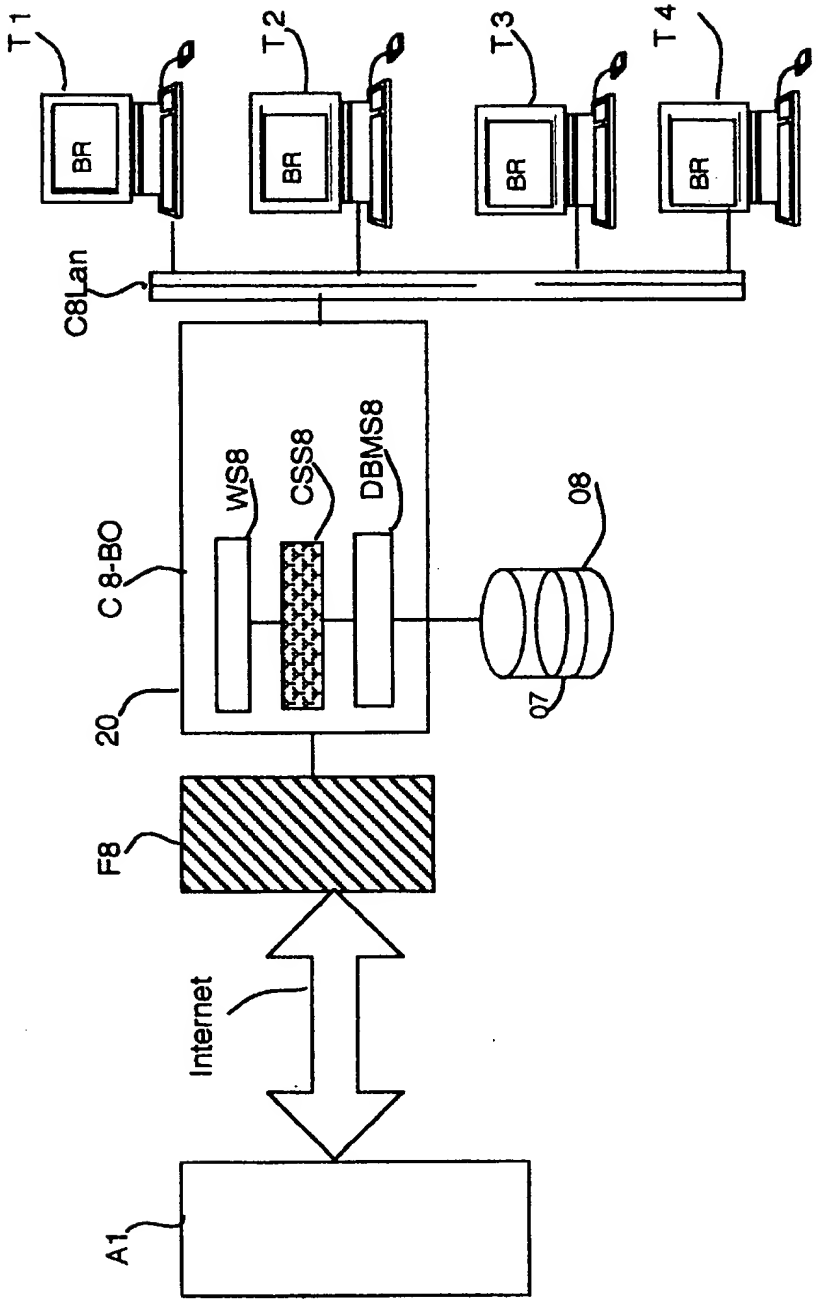

T1
T2
T3
T4
BR
BR
BR
BR
C8Lan
WS8
CSS8
DBMS8
08
C 8-BO
20
07
F8
Internet
A1

Fig. 4

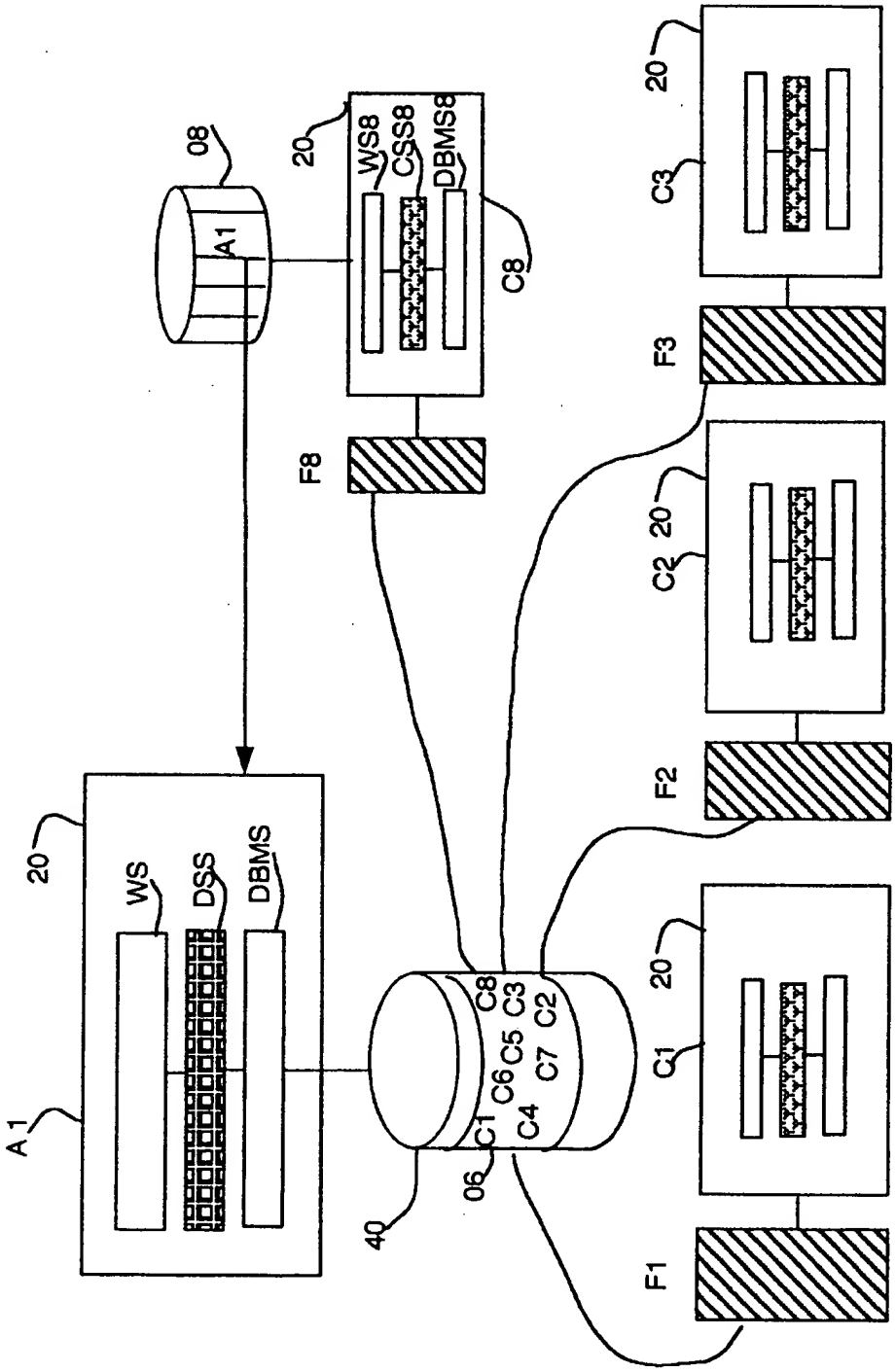

08
20
WS8
CSS8
DBMS8
A1
C8
F8
20
C3
F3
20
C2
F2
20
C1
F1
20
WS
DSS
DBMS
A1
C8
C3
C2
C5
C7
C6
C4
C1
06
40

Fig. 5

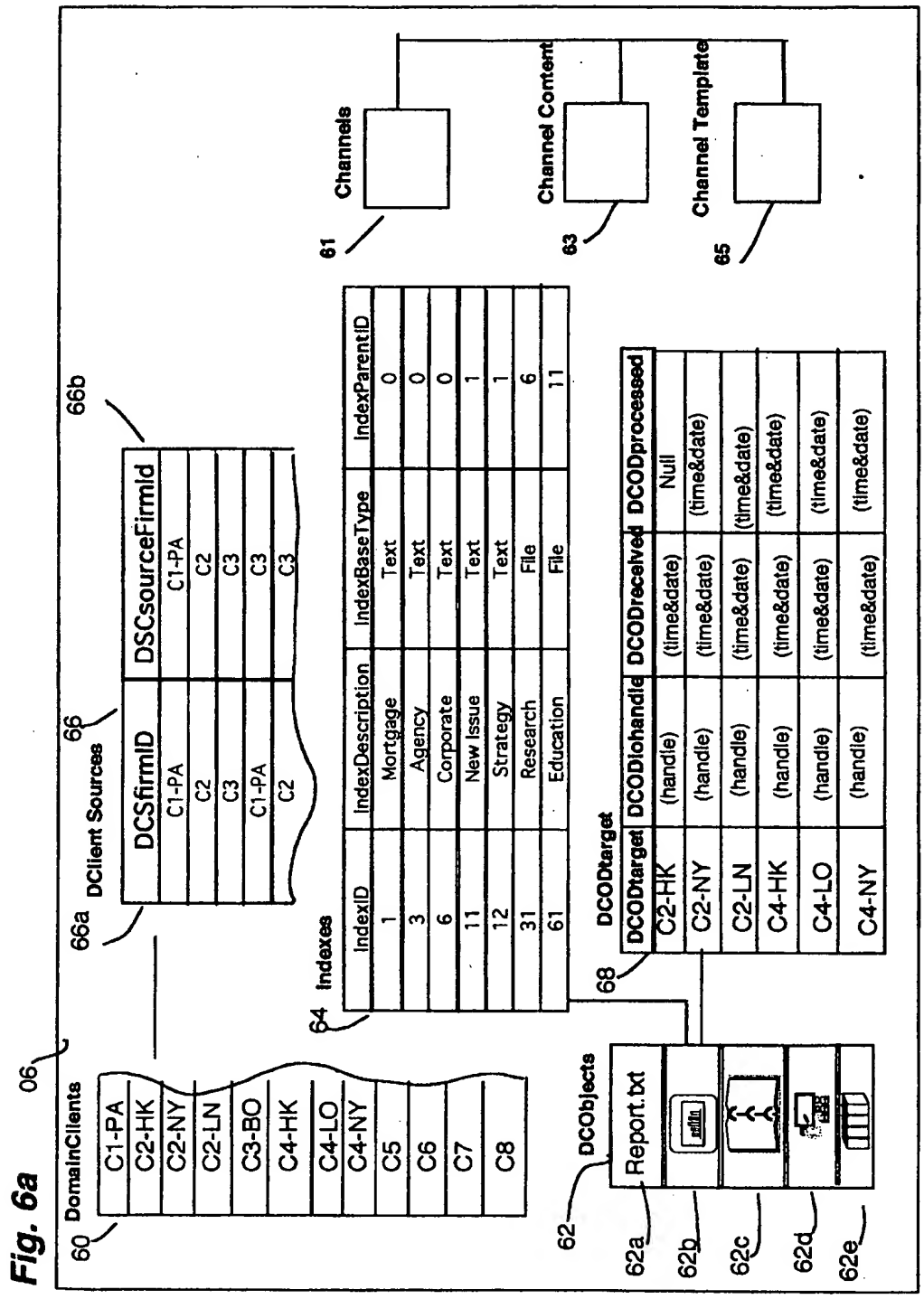


Fig. 6a
06
DomainClients
60
C1-PA
C2-HK
C2-NY
C2-LN
C3-BO
C4-HK
C4-LO
C4-NY
C5
C6
C7
C8
DClient Sources
66
66a
DCSfirmID
DCSsourceFirmId
66b
C1-PA
C2
C3
C1-PA
C2
C1-PA
C2
C3
C3
C3
Indexes
64
IndexID
IndexDescription
IndexBaseType
IndexParentID
1
Mortgage
Text
0
3
Agency
Text
0
6
Corporate
Text
0
11
New Issue
Text
1
12
Strategy
Text
1
31
Research
File
6
61
Education
File
11
Channels
61
Channel Content
63
Channel Template
65
DCOTarget
68
DCOTarget
DCODiohandle
DCODreceived
DCODprocessed
C2-HK
(handle)
(time&date)
Null
C2-NY
(handle)
(time&date)
(time&date)
C2-LN
(handle)
(time&date)
(time&date)
C4-HK
(handle)
(time&date)
(time&date)
C4-LO
(handle)
(time&date)
(time&date)
C4-NY
(handle)
(time&date)
(time&date)
DCObjects
62
Report.txt
62a
62b
62c
62d
62e

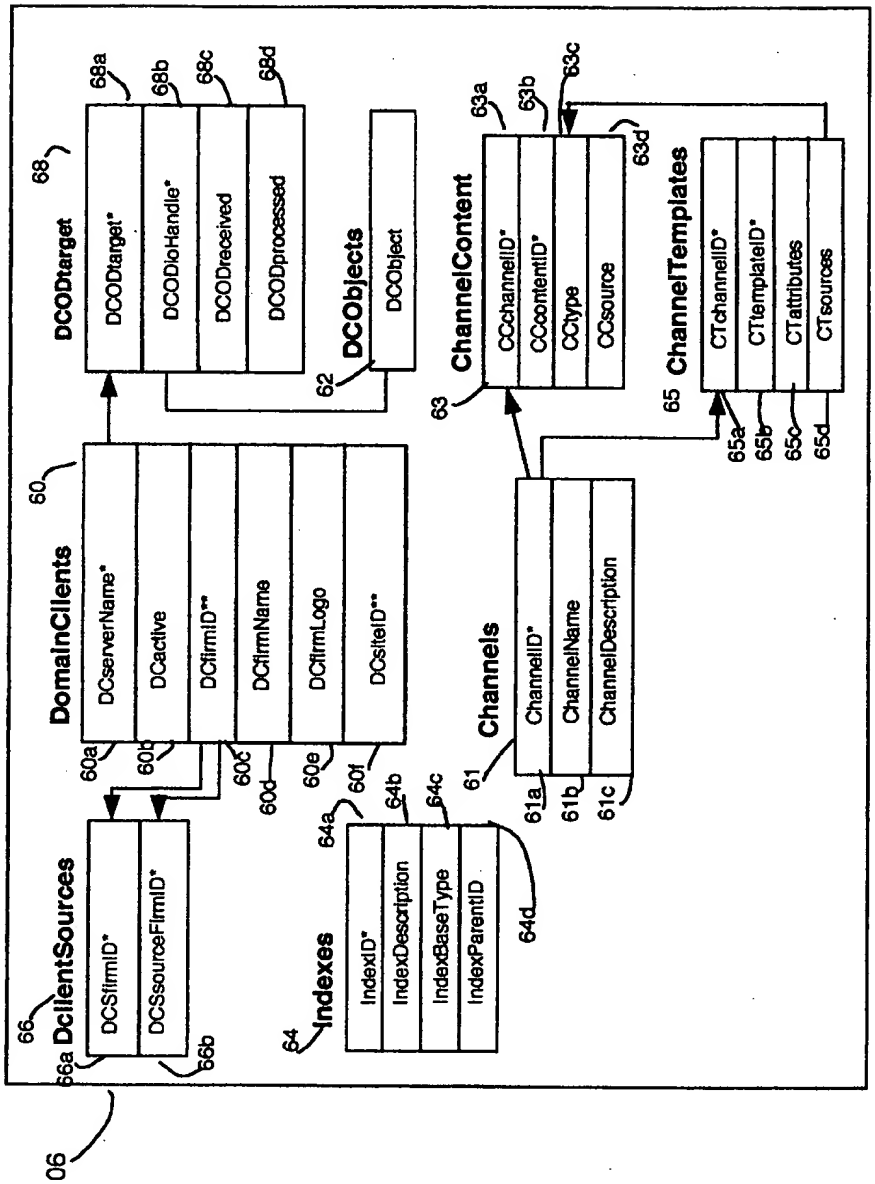

DCODtarget
DCODtarget*
DCODioHandle*
DCODreceived
DCODprocessed
DCObjects
DCObject
ChannelContent
CCchannelID*
CCcontentID*
CCtype
CCsource
ChannelTemplates
CTchannelID*
CTtemplateID*
CTattributes
CTsources
DomainClients
DCserverName*
DCactive
DCfirmID**
DCfirmName
DCfirmLogo
DCsiteID**
Channels
ChannelID*
ChannelName
ChannelDescription
DclientSources
DCSfirmID*
DCSsourceFirmID*
Indexes
IndexID*
IndexDescription
IndexBaseType
IndexParentID
68
68a
68b
68c
68d
62
63
63a
63b
63c
63d
65
65a
65b
65c
65d
60
60a
60b
60c
60d
60e
60f
61
61a
61b
61c
64a
64b
64c
64d
66
66a
66b
64
06

Fig. 6b

Fig. 7a

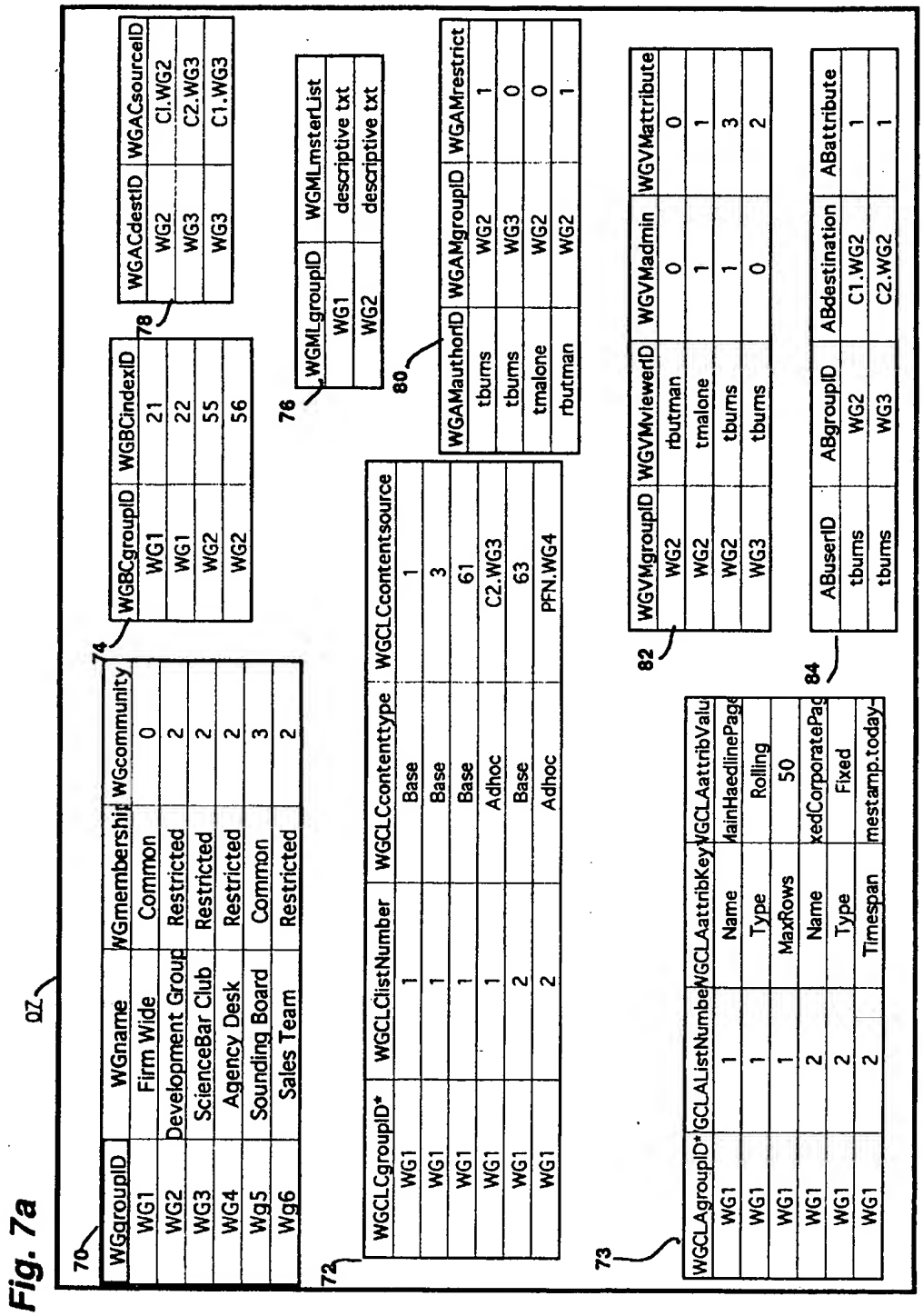

70
WGgroupID | WGname | WGmembership | WGcommunity
WG1 | Firm Wide | Common | 0
WG2 | Development Group | Restricted | 2
WG3 | ScienceBar Club | Restricted | 2
WG4 | Agency Desk | Restricted | 2
Wg5 | Sounding Board | Common | 3
Wg6 | Sales Team | Restricted | 2
72
WGCLCgroupID* | WGCLClistNumber | WGCLCcontenttype | WGCLCcontentsource
WG1 | 1 | Base | 1
WG1 | 1 | Base | 3
WG1 | 1 | Base | 61
WG1 | 1 | Adhoc | C2.WG3
WG1 | 2 | Base | 63
WG1 | 2 | Adhoc | PFN.WG4
73
WGCLAgroupID* | GCLAListNumbe | WGCLAattribKey | WGCLAattribValu
WG1 | 1 | Name | MainHaedlinePage
WG1 | 1 | Type | Rolling
WG1 | 1 | MaxRows | 50
WG1 | 2 | Name | xedCorporatePag
WG1 | 2 | Type | Fixed
WG1 | 2 | Timespan | mestamp.today-
74
WGBCgroupID | WGBCindexID
WG1 | 21
WG1 | 22
WG2 | 55
WG2 | 56
78
WGACdestID | WGACsourceID
WG2 | C1.WG2
WG3 | C2.WG3
WG3 | C1.WG3
76
WGMLgroupID | WGMLmsterList
WG1 | descriptive txt
WG2 | descriptive txt
80
WGAMauthorID | WGAMgroupID | WGAMrestrict
tburns | WG2 | 1
tburns | WG3 | 0
tmalone | WG2 | 0
rbutman | WG2 | 1
82
WGVMgroupID | WGVMviewerID | WGVMadmin | WGVMattribute
WG2 | rbutman | 0 | 0
WG2 | tmalone | 1 | 1
WG2 | tburns | 1 | 3
WG3 | tburns | 0 | 2
84
ABuserID | ABgroupID | ABdestination | ABattribute
tburns | WG2 | C1.WG2 | 1
tburns | WG3 | C2.WG2 | 1

Fig. 7a, ct'd

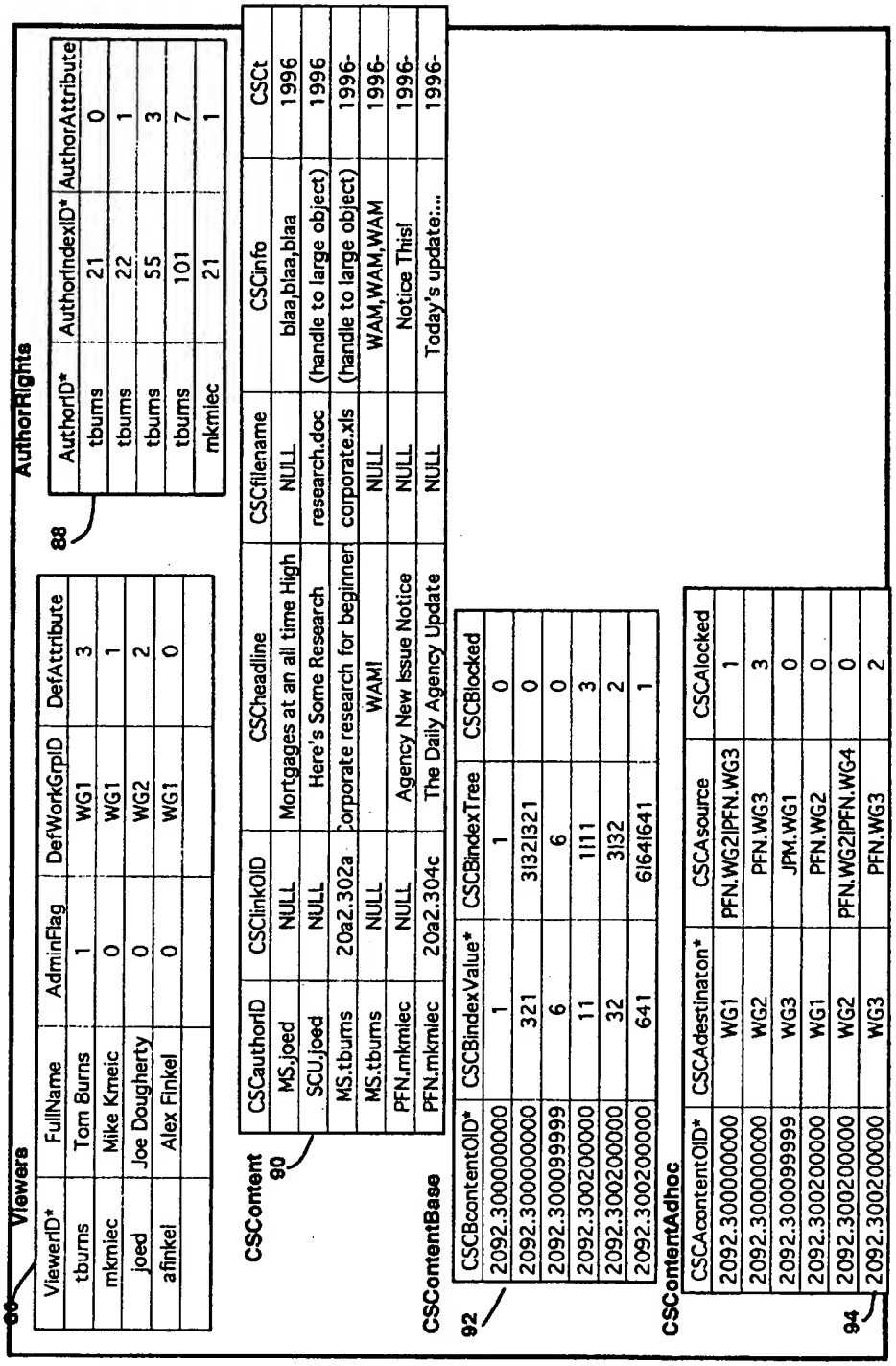

Viewers
ViewerID* | FullName | AdminFlag | DefWorkGrpID | DefAttribute
tburns | Tom Burns | 1 | WG1 | 3
mkmiec | Mike Kmeic | 0 | WG1 | 1
joed | Joe Dougherty | 0 | WG2 | 2
afinkel | Alex Finkel | 0 | WG1 | 0
AuthorRights
AuthorID* | AuthorIndexID* | AuthorAttribute
tburns | 21 | 0
tburns | 22 | 1
tburns | 55 | 3
tburns | 101 | 7
mkmiec | 21 | 1
CSContent
CSCauthorID | CSClinkOID | CSCheadline | CSCfilename | CSCinfo | CSCt
MS.joed | NULL | Mortgages at an all time High | NULL | blaa,blaa,blaa | 1996
SCU.joed | NULL | Here's Some Research | research.doc | (handle to large object) | 1996
MS.tburns | 20a2.302a | Corporate research for beginner | corporate.xls | (handle to large object) | 1996-
MS.tburns | NULL | WAM! | NULL | WAM,WAM,WAM | 1996-
PFN.mkmiec | NULL | Agency New Issue Notice | NULL | Notice This! | 1996-
PFN.mkmiec | 20a2.304c | The Daily Agency Update | NULL | Today's update:... | 1996-
CSContentBase
CSCBcontentOID* | CSCBindexValue* | CSCBindexTree | CSCBlocked
2092.300000000 | 1 | 1 | 0
2092.300000000 | 321 | 3132\321 | 0
2092.300099999 | 6 | 6 | 0
2092.300200000 | 11 | 111 | 3
2092.300200000 | 32 | 3132 | 2
2092.300200000 | 641 | 61641641 | 1
CSContentAdhoc
CSCAcontentOID* | CSCAdestinaton* | CSCAsource | CSCAlocked
2092.300000000 | WG1 | PFN.WG2\PFN.WG3 | 1
2092.300000000 | WG2 | PFN.WG3 | 3
2092.300099999 | WG3 | JPM.WG1 | 0
2092.300200000 | WG1 | PFN.WG2 | 0
2092.300200000 | WG2 | PFN.WG2\PFN.WG4 | 0
2092.300200000 | WG3 | PFN.WG3 | 2
00
02
88
90
92
94

Fig. 7a, ct'd

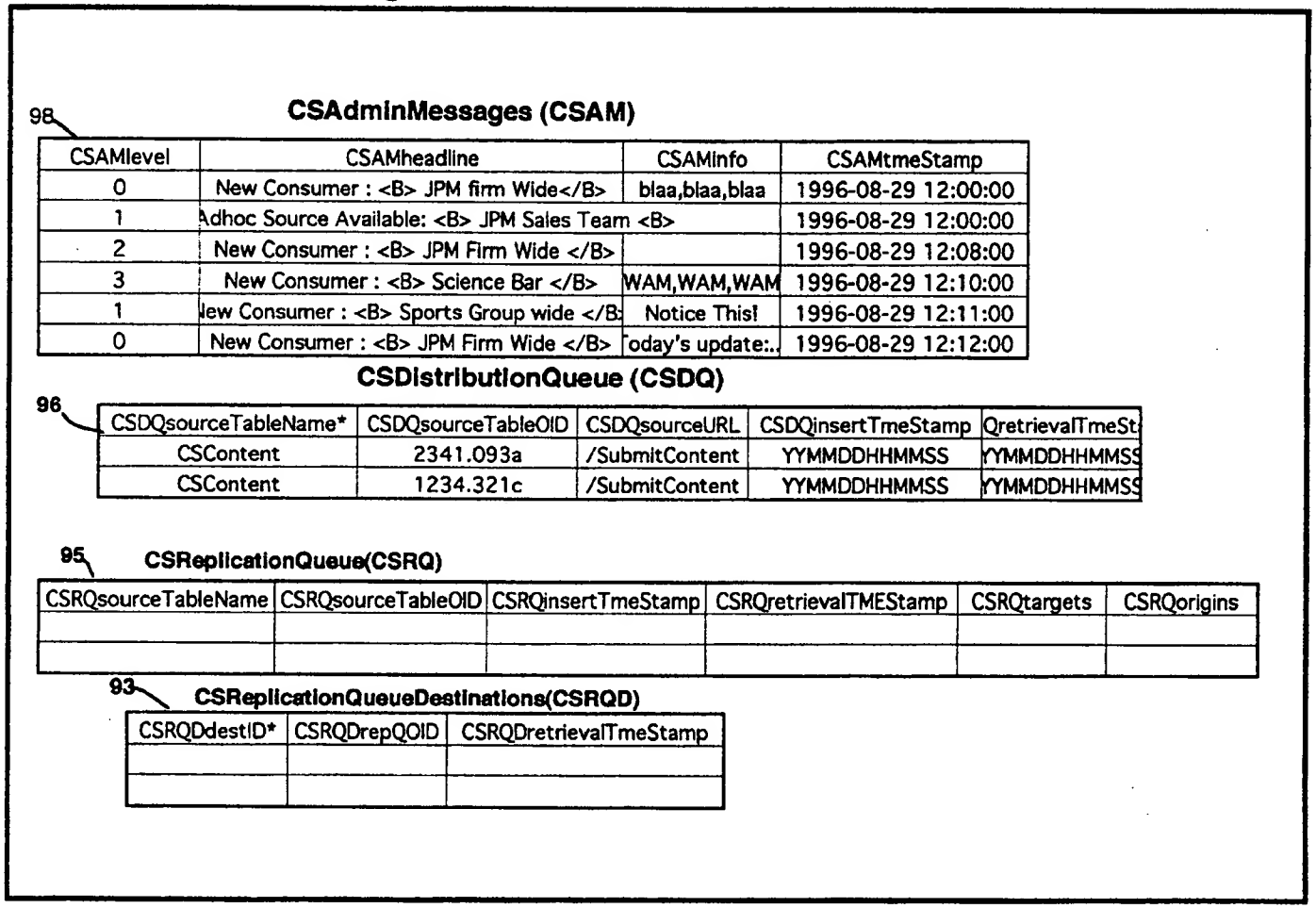
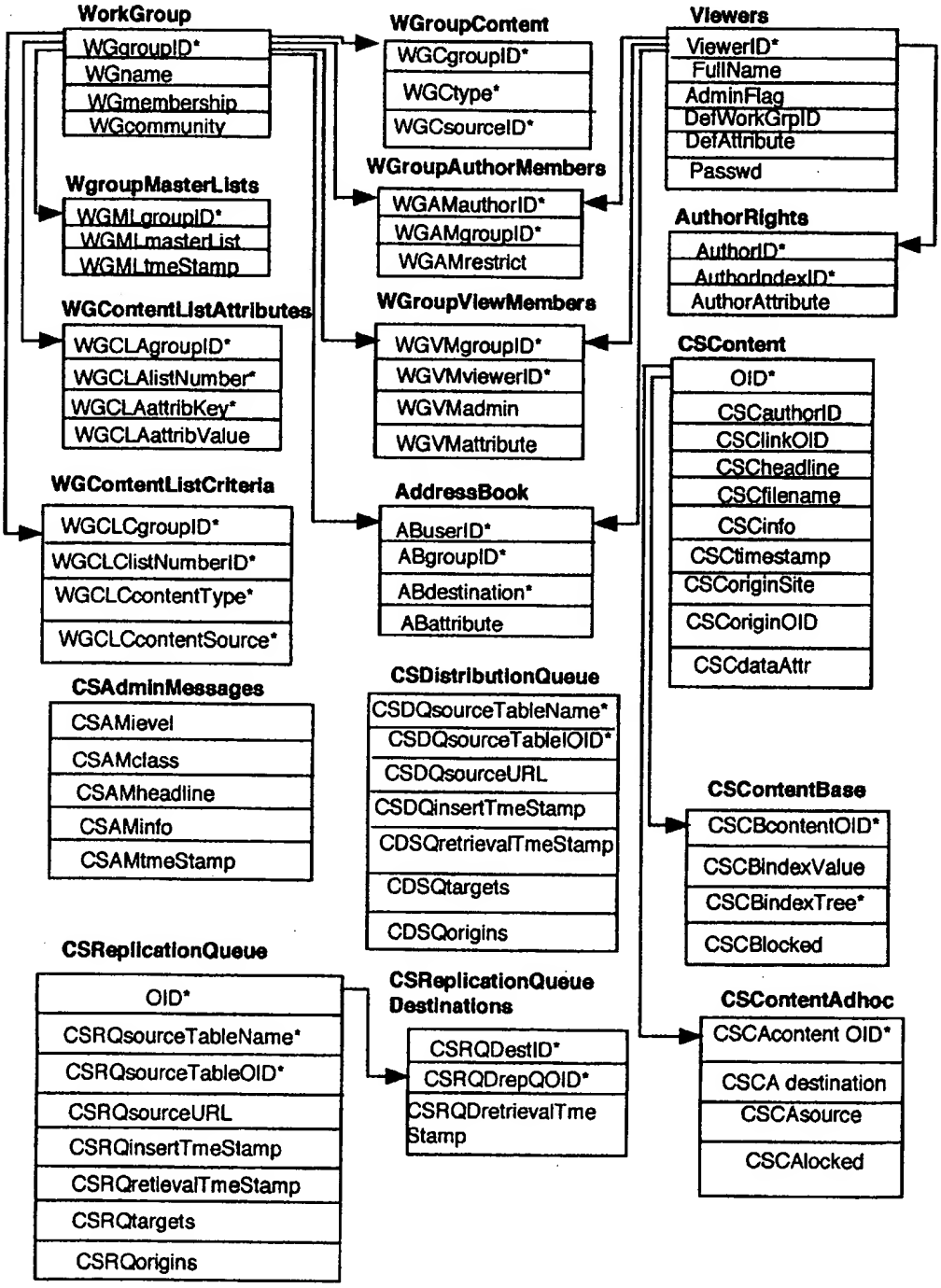

97
98
CSAdminMessages (CSAM)
CSAMlevel | CSAMheadline | CSAMinfo | CSAMtmeStamp
0 | New Consumer : <B> JPM firm Wide</B> | blaa,blaa,blaa | 1996-08-29 12:00:00
1 | Adhoc Source Available: <B> JPM Sales Team <B> | | 1996-08-29 12:00:00
2 | New Consumer : <B> JPM Firm Wide </B> | | 1996-08-29 12:08:00
3 | New Consumer : <B> Science Bar </B> | WAM,WAM,WAM | 1996-08-29 12:10:00
1 | New Consumer : <B> Sports Group wide </B> | Notice This! | 1996-08-29 12:11:00
0 | New Consumer : <B> JPM Firm Wide </B> | Today's update:.. | 1996-08-29 12:12:00
CSDistributionQueue (CSDQ)
96
CSDQsourceTableName* | CSDQsourceTableOID | CSDQsourceURL | CSDQinsertTmeStamp | QretrievalTmeSt
CSContent | 2341.093a | /SubmitContent | YYMMDDHHMMSS | YYMMDDHHMMSS
CSContent | 1234.321c | /SubmitContent | YYMMDDHHMMSS | YYMMDDHHMMSS
95
CSReplicationQueue(CSRQ)
CSRQsourceTableName | CSRQsourceTableOID | CSRQinsertTmeStamp | CSRQretrievalTMEStamp | CSRQtargets | CSRQorigins
93
CSReplicationQueueDestinations(CSRQD)
CSRQDdestID* | CSRQDrepQOID | CSRQDretrievalTmeStamp

Fig. 7b


WorkGroup
WGgroupID*
WGname
WGmembership
WGcommunity
WGroupContent
WGCgroupID*
WGCtype*
WGCsourceID*
Viewers
ViewerID*
FullName
AdminFlag
DefWorkGrpID
DefAttribute
Passwd
WGroupAuthorMembers
WGAMauthorID*
WGAMgroupID*
WGAMrestrict
AuthorRights
AuthorID*
AuthorIndexID*
AuthorAttribute
WgroupMasterLists
WGMLgroupID*
WGMLmasterList
WGMLtmeStamp
WGContentListAttributes
WGCLAgroupID*
WGCLAlistNumber*
WGCLAattribKey*
WGCLAattribValue
WGroupViewMembers
WGVMgroupID*
WGVMviewerID*
WGVMadmin
WGVMattribute
CSContent
OID*
CSCauthorID
CSClinkOID
CSCheadline
CSCfilename
CSCinfo
CSCtimestamp
CSCoriginSite
CSCoriginOID
CSCdataAttr
WGContentListCriteria
WGCLCgroupID*
WGCLClistNumberID*
WGCLCcontentType*
WGCLCcontentSource*
AddressBook
ABuserID*
ABgroupID*
ABdestination*
ABattribute
CSAdminMessages
CSAMlevel
CSAMclass
CSAMheadline
CSAMinfo
CSAMtmeStamp
CSDistributionQueue
CSDQsourceTableName*
CSDQsourceTableIOID*
CSDQsourceURL
CSDQinsertTmeStamp
CDSQretrievalTmeStamp
CDSQtargets
CDSQorigins
CSContentBase
CSCBcontentOID*
CSCBindexValue
CSCBindexTree*
CSCBlocked
CSReplicationQueue
OID*
CSRQsourceTableName*
CSRQsourceTableOID*
CSRQsourceURL
CSRQinsertTmeStamp
CSRQretlevalTmeStamp
CSRQtargets
CSRQorigins
CSReplicationQueue Destinations
CSRQDestID*
CSRQDrepQOID*
CSRQDretrievalTme Stamp
CSContentAdhoc
CSCAcontent OID*
CSCA destination
CSCAsource
CSCAlocked

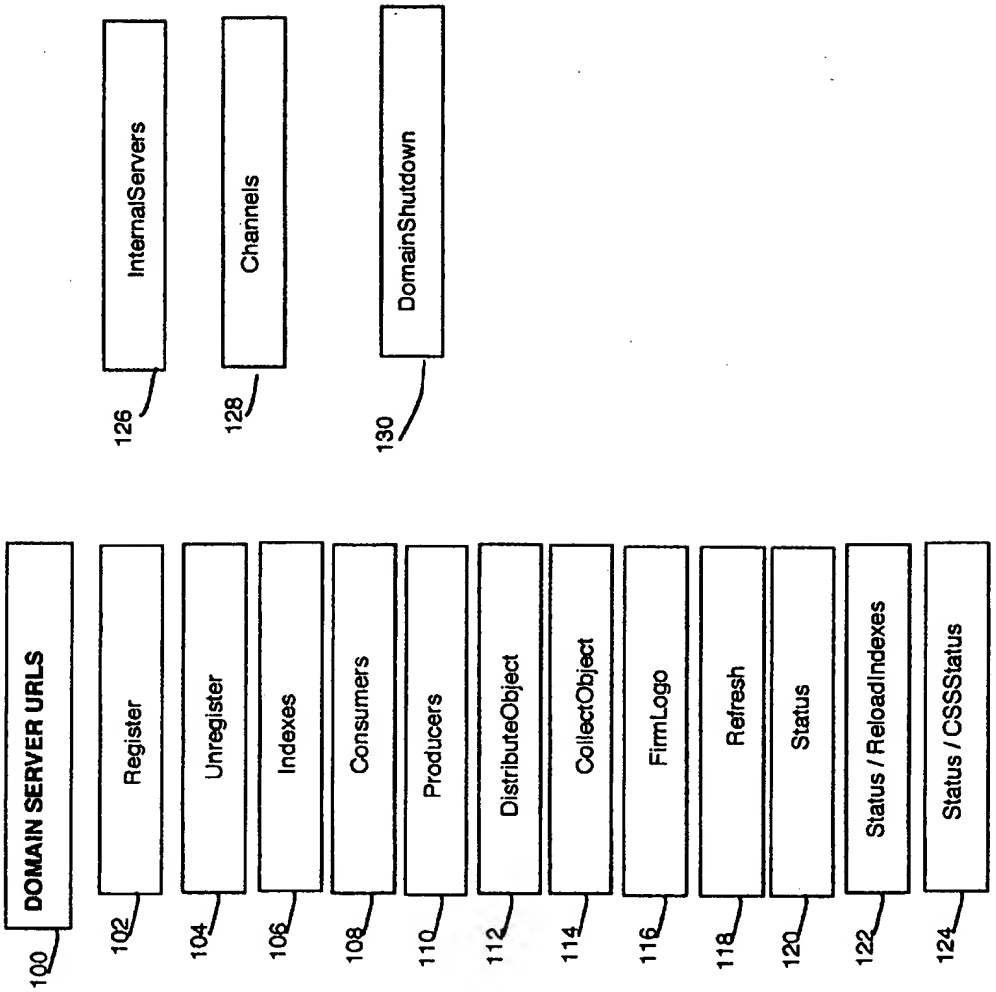

DOMAIN SERVER URLS
100
Register
102
Unregister
104
Indexes
106
Consumers
108
Producers
110
DistributeObject
112
CollectObject
114
FirmLogo
116
Refresh
118
Status
120
Status / ReloadIndexes
122
Status / CSSStatus
124
InternalServers
126
Channels
128
DomainShutdown
130

Fig. 8a

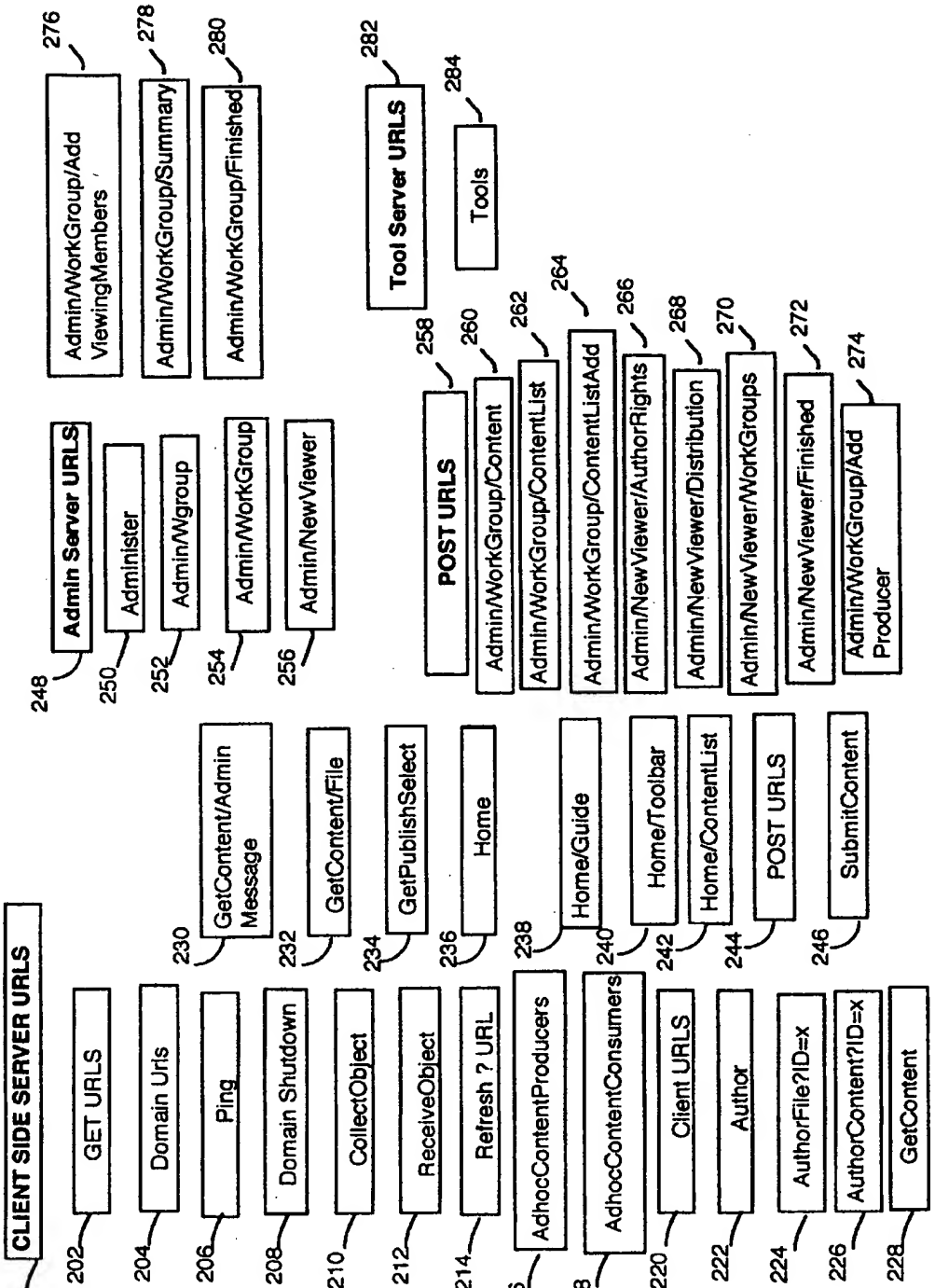

CLIENT SIDE SERVER URLS
200
GET URLS
202
Domain Urls
204
Ping
206
Domain Shutdown
208
CollectObject
210
ReceiveObject
212
Refresh ? URL
214
AdhocContentProducers
216
AdhocContentConsumers
218
Client URLS
220
Author
222
AuthorFile?ID=x
224
AuthorContent?ID=x
226
GetContent
228
GetContent/Admin Message
230
GetContent/File
232
GetPublishSelect
234
Home
236
Home/Guide
238
Home/Toolbar
240
Home/ContentList
242
POST URLS
244
SubmitContent
246
Admin Server URLS
248
Administer
250
Admin/Wgroup
252
Admin/WorkGroup
254
Admin/NewViewer
256
Admin/WorkGroup/Add ViewingMembers
276
Admin/WorkGroup/Summary
278
Admin/WorkGroup/Finished
280
POST URLS
258
Admin/WorkGroup/Content
260
Admin/WorkGroup/ContentList
262
Admin/WorkGroup/ContentListAdd
264
Admin/NewViewer/AuthorRights
266
Admin/NewViewer/Distribution
268
Admin/NewViewer/WorkGroups
270
Admin/NewViewer/Finished
272
Admin/WorkGroup/Add Producer
274
Tool Server URLS
282
Tools
284

Fig. 8b

Fig. 9

DCserverName	DCactive	DCfirmID	DCfirmName	DCfirmLogo
http://helios.C2-HK.com:8888	1	C2	Company2	C2logo.jpg
http://hellos.C2-NY.com:8889	1	C2	Company2	C2logo.gif
http://myserver.C3.com:80	1	C3	Company3	C3logo.gif
http://www.pfn.com:80	0	PFN	PFN, Inc.	PFNlogo.gif

Fig. 10

DCSfirmID	DCSsourceFirmID
C1	C1
C2	C1
C2	C2
C2	C3

Fig. 11

DCOobject
<url><origins><targets><data>

Fig. 12

DCODtarget*	DCODIoHandle	DCODreceived	DCODprocessed
http://www.pfn.com:8213	1010982029384	1996-08-29 12:00:00	1996-08-29 12:00:01
http://www.pfn.com:8218	1010982029384	1996-08-29 12:00:00	1996-08-29 12:00:01
http://my.sever.com	1010982029385	1996-08-29 12:05:00	NULL

Fig. 13

IndexID*	IndexDescription	IndexBaseType	IndexParentID
1	Engineering	Text	0
3	Development	Text	0
6	Hardware	Text	0
11	New Products	Text	1
12	Strategy	Text	1
31	Software	Text	3
61	Research	Text	6
111	QA	Text	11

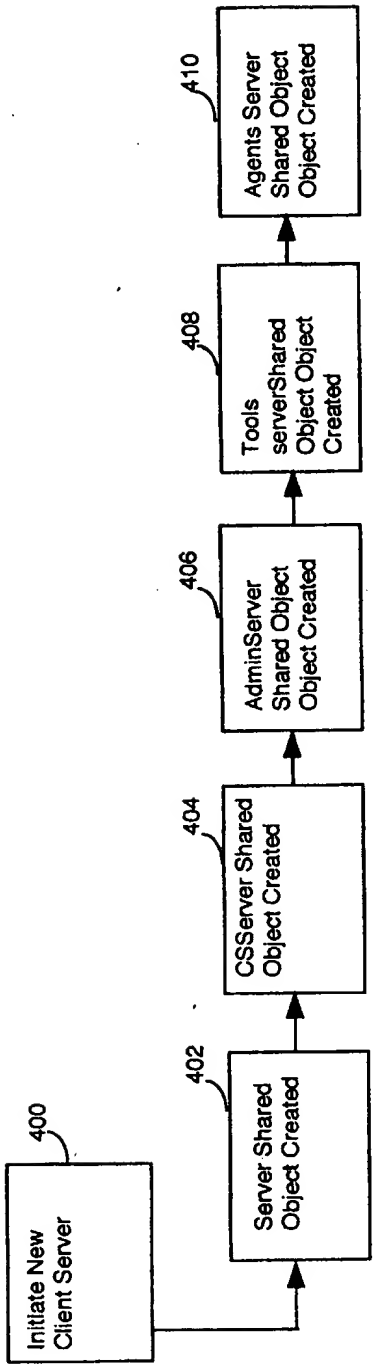

Initiate New Client Server
400
Server Shared Object Created
402
CSServer Shared Object Created
404
AdminServer Shared Object Object Created
406
Tools serverShared Object Object Created
408
Agents Server Shared Object Object Created
410

Fig. 14

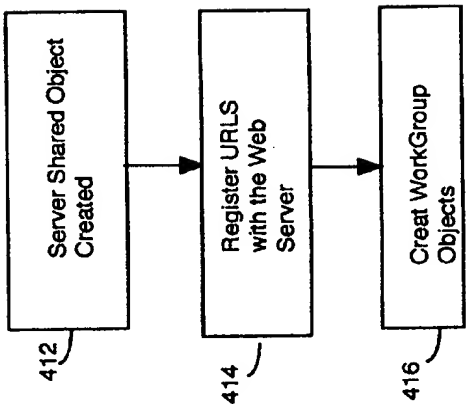

Server Shared Object Created
412
Register URLS with the Web Server
414
Creat WorkGroup Objects
416

Fig. 15

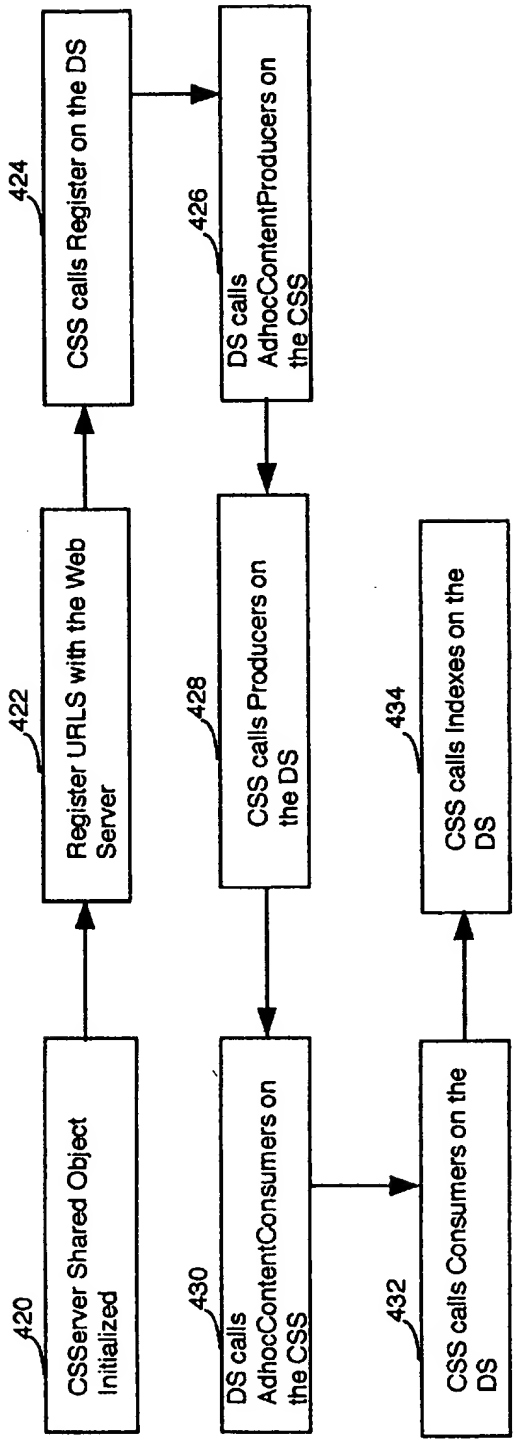

CSServer Shared Object Initialized
420
Register URLS with the Web Server
422
CSS calls Register on the DS
424
DS calls AdhocContentConsumers on the CSS
430
CSS calls Producers on the DS
428
DS calls AdhocContentProducers on the CSS
426
CSS calls Consumers on the DS
432
CSS calls Indexes on the DS
434

Fig. 16

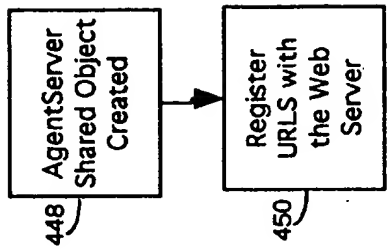


Fig. 19

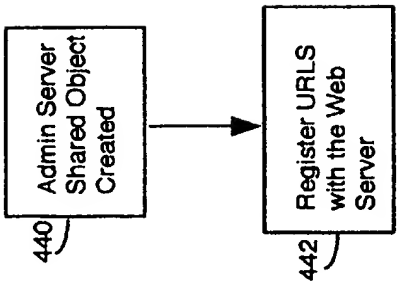


Fig. 17

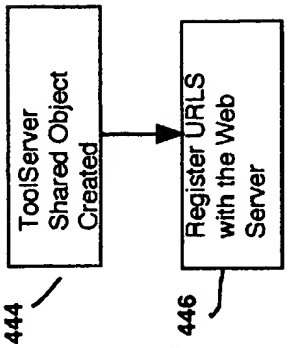


Fig. 18

Fig. 20a

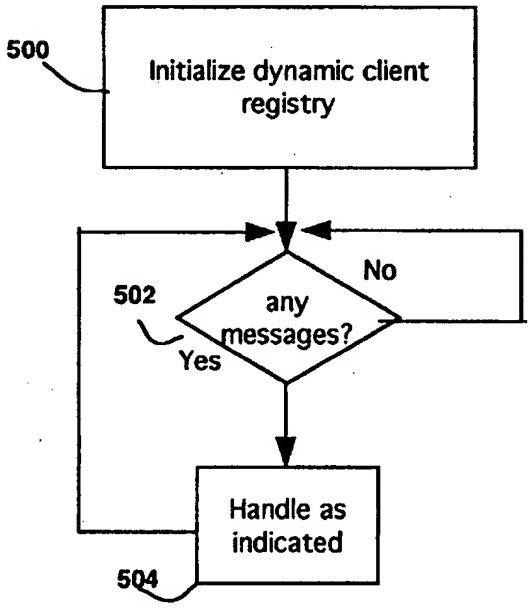

500
Initialize dynamic client registry
502
any messages?
No
Yes
Handle as indicated
504

Fig. 20b

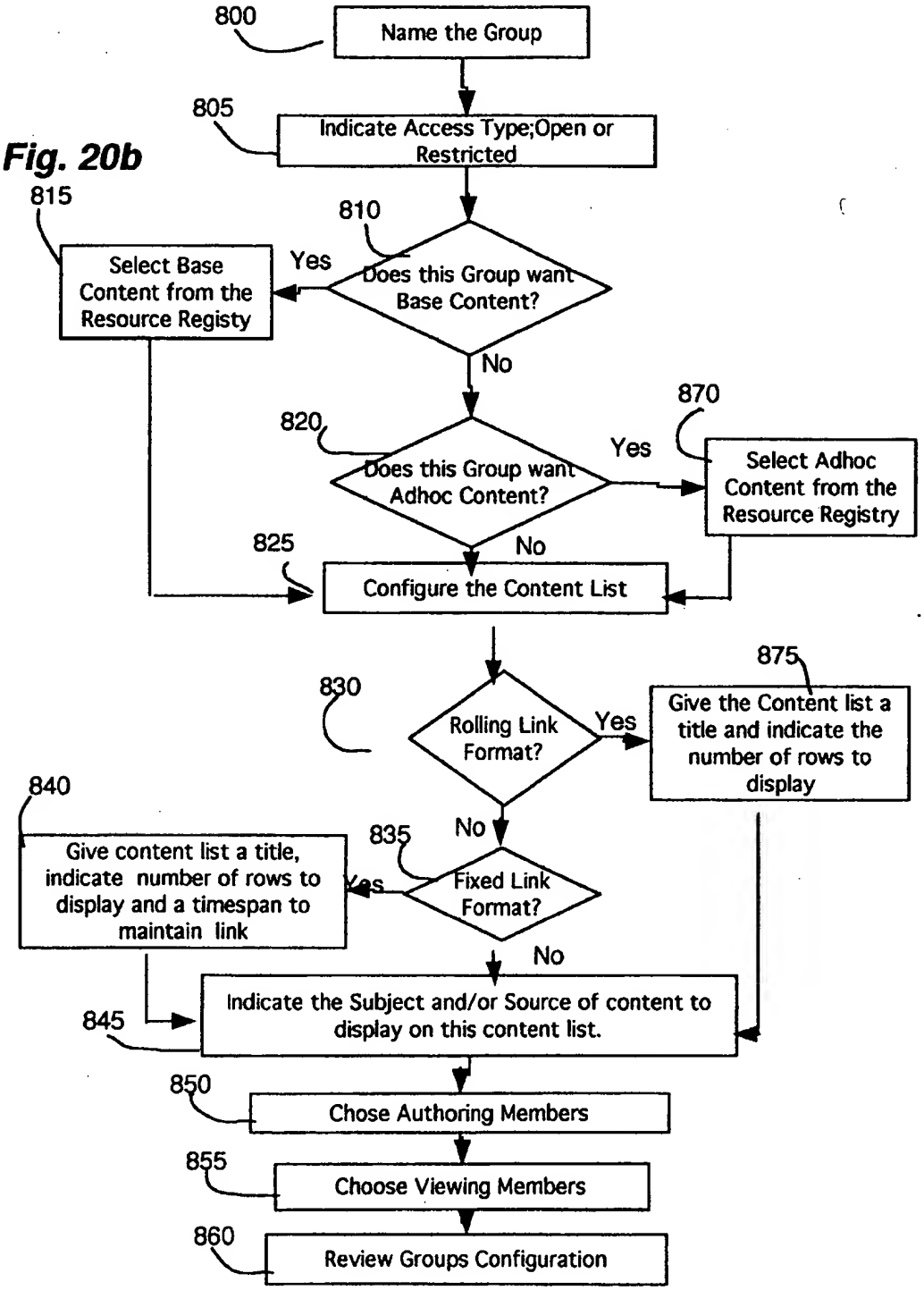
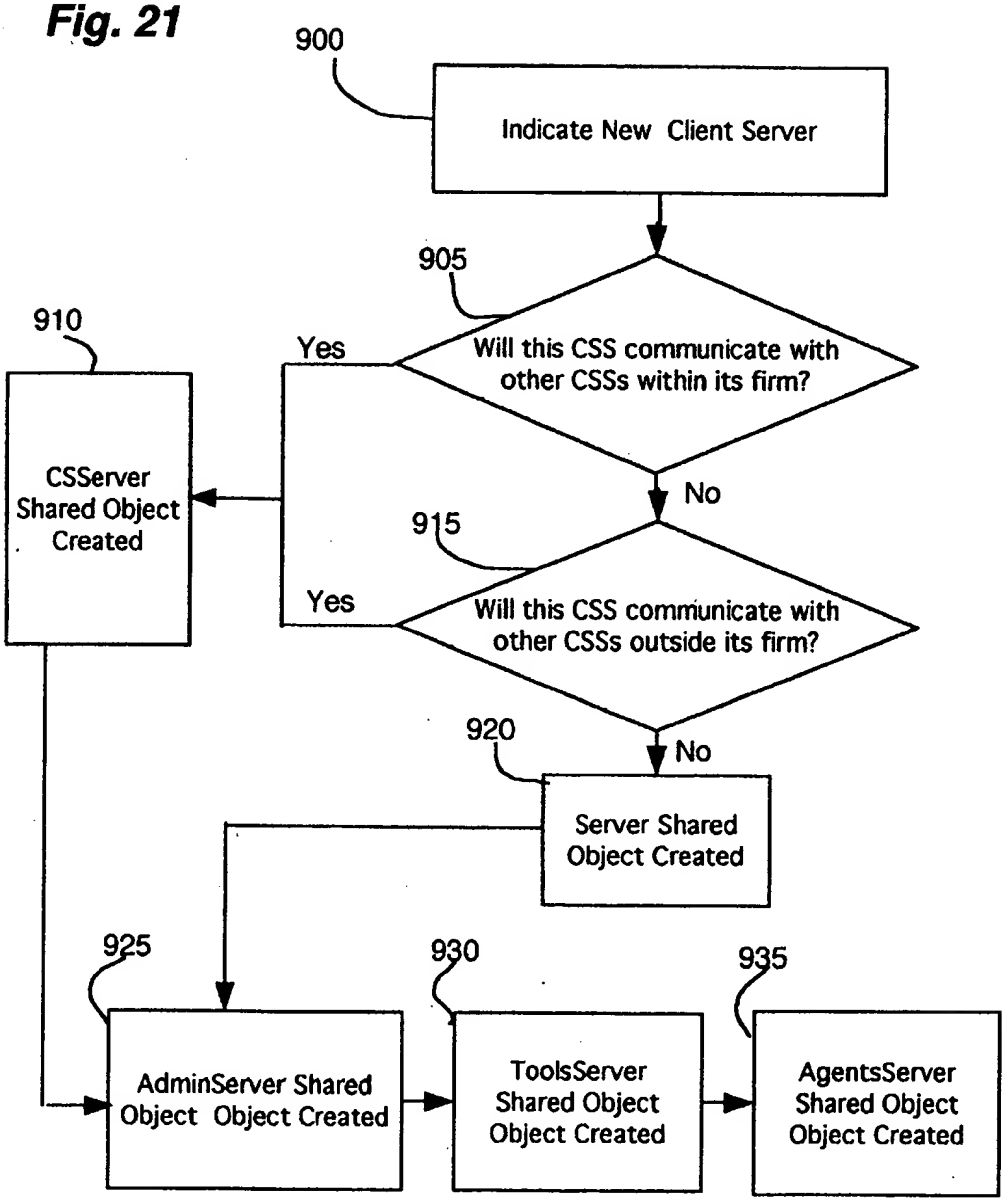

800
Name the Group
805
Indicate Access Type;Open or Restricted
810
Does this Group want Base Content?
Yes
815
Select Base Content from the Resource Registy
No
820
Does this Group want Adhoc Content?
Yes
870
Select Adhoc Content from the Resource Registry
No
825
Configure the Content List
830
Rolling Link Format?
Yes
875
Give the Content list a title and indicate the number of rows to display
No
835
Fixed Link Format?
Yes
840
Give content list a title, indicate number of rows to display and a timespan to maintain link
No
845
Indicate the Subject and/or Source of content to display on this content list.
850
Chose Authoring Members
855
Choose Viewing Members
860
Review Groups Configuration

Fig. 21


900
Indicate New Client Server
905
Will this CSS communicate with other CSSs within its firm?
Yes
No
910
CSServer Shared Object Created
915
Will this CSS communicate with other CSSs outside its firm?
Yes
No
920
Server Shared Object Created
925
AdminServer Shared Object Object Created
930
ToolsServer Shared Object Object Created
935
AgentsServer Shared Object Object Created

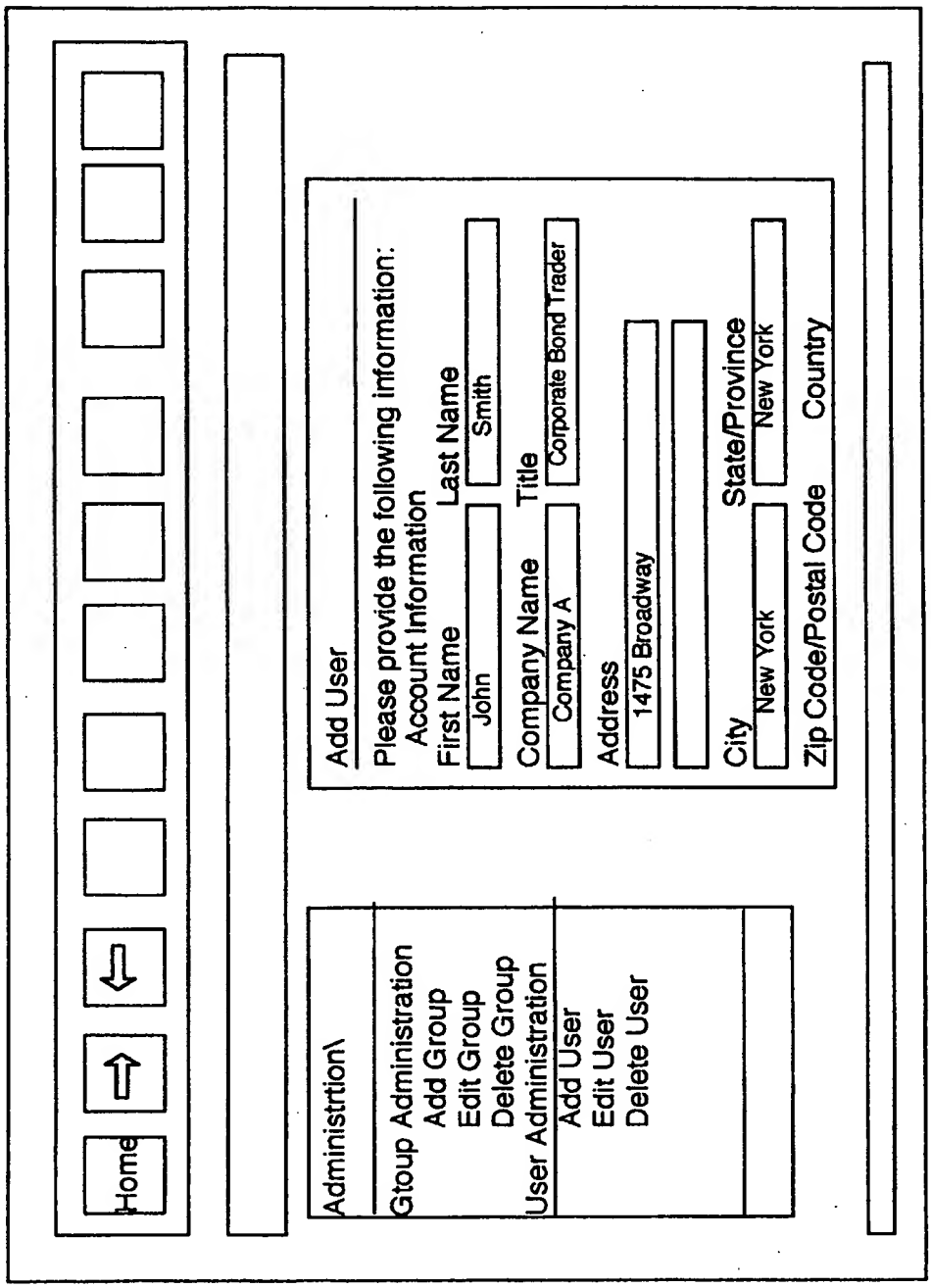

Home
Administrtion\
Gtoup Administration
Add Group
Edit Group
Delete Group
User Administration
Add User
Edit User
Delete User
Add User
Please provide the following information:
Account Information
First Name
John
Last Name
Smith
Company Name
Company A
Title
Corporate Bond Trader
Address
1475 Broadway
City
New York
State/Province
New York
Zip Code/Postal Code
Country

Fig. 22

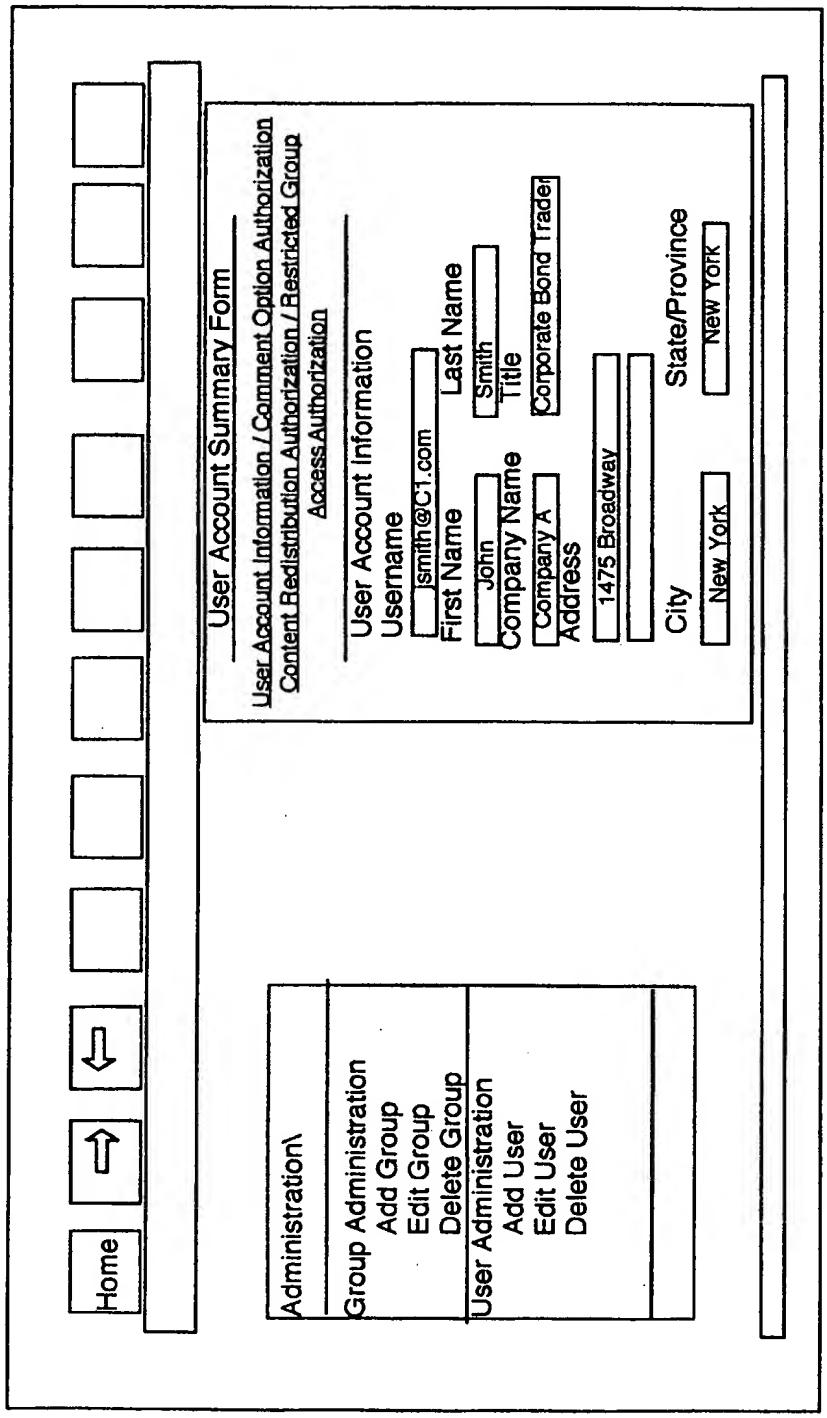

Home
Administration\
Group Administration
Add Group
Edit Group
Delete Group
User Administration
Add User
Edit User
Delete User
User Account Summary Form
User Account Information / Comment Option Authorization
Content Redistribution Authorization / Restricted Group
Access Authorization
User Account Information
Username
jsmith@C1.com
First Name
John
Last Name
Smith
Company Name
Company A
Title
Corporate Bond Trader
Address
1475 Broadway
City
New York
State/Province
New York

Fig. 23

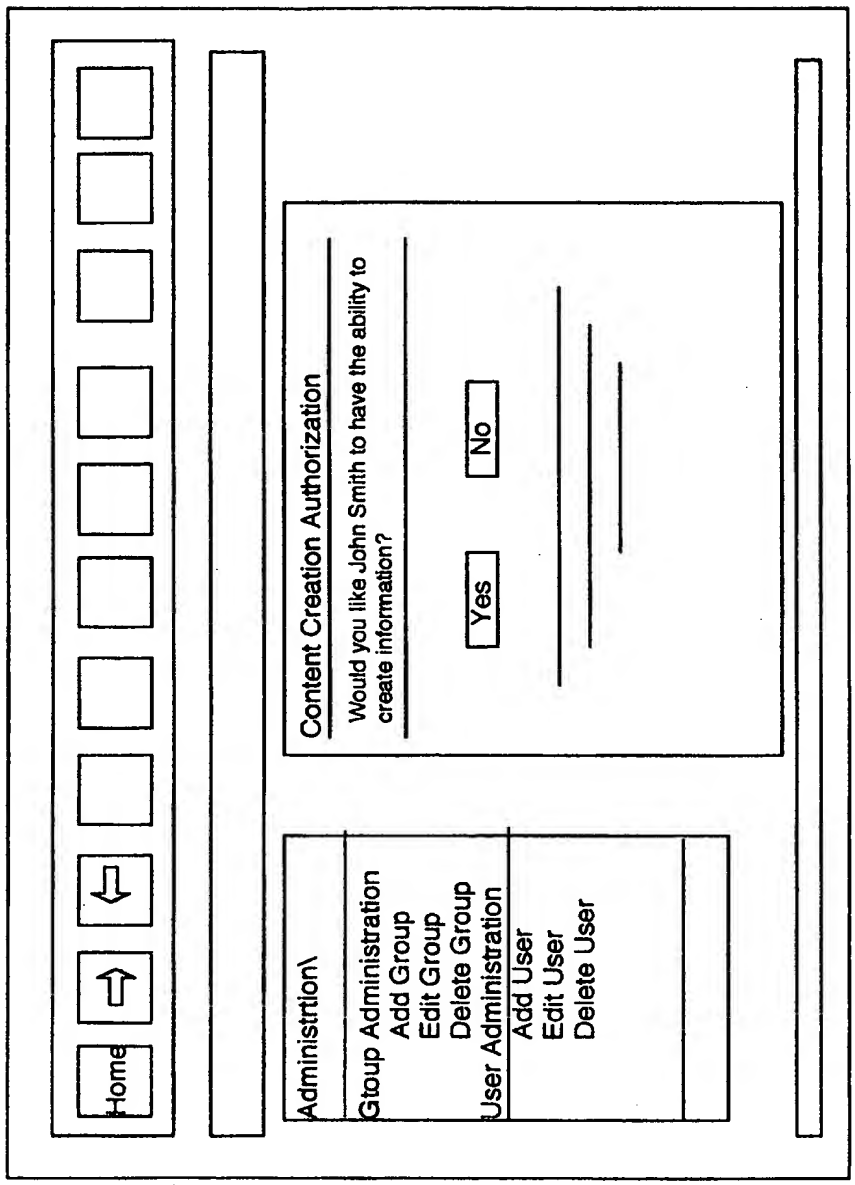
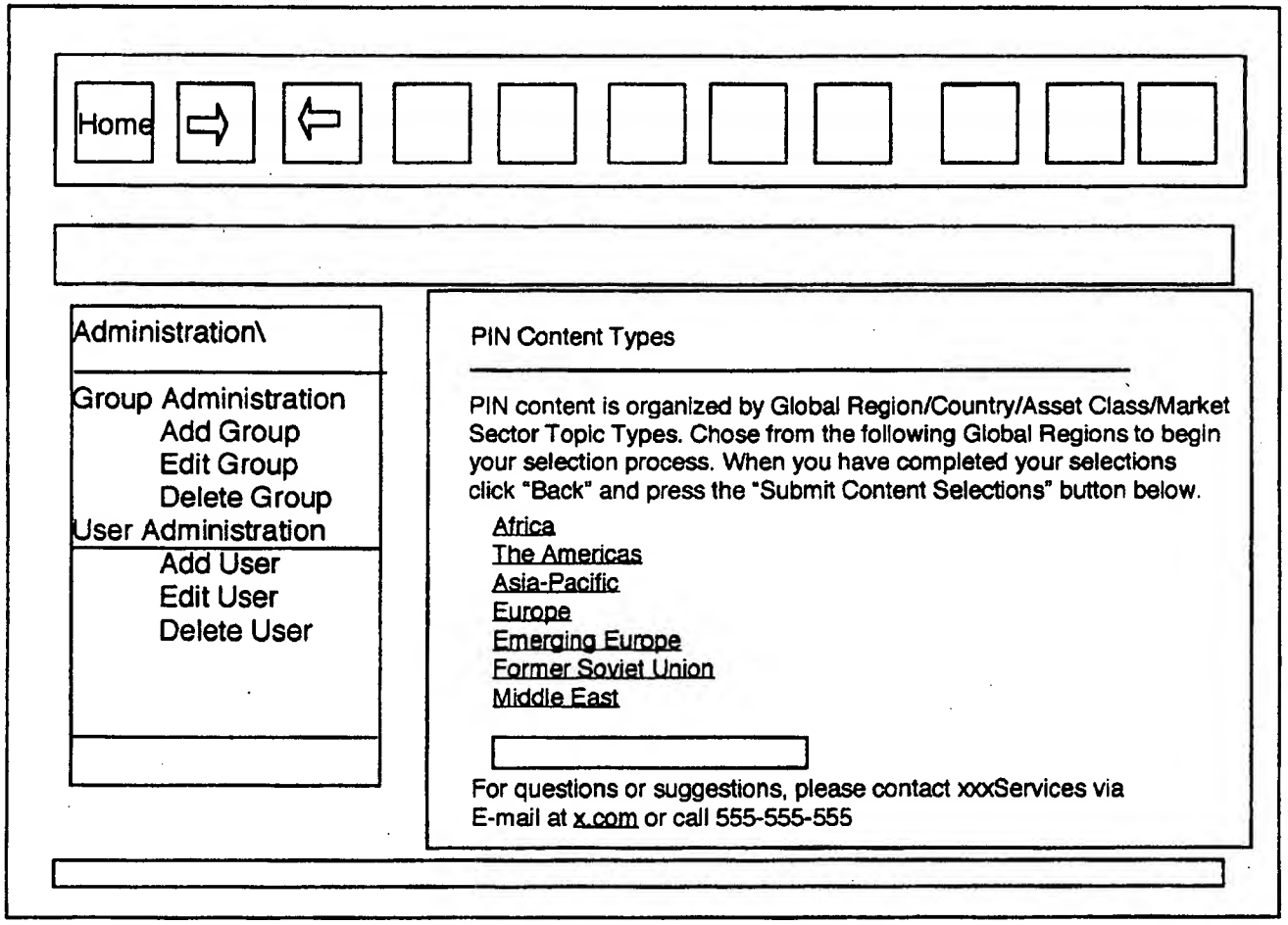

Home
Administrtion\
Gtoup Administration
Add Group
Edit Group
Delete Group
User Administration
Add User
Edit User
Delete User
Content Creation Authorization
Would you like John Smith to have the ability to create information?
Yes
No

Fig. 24

Fig. 25a


Home
Administration\
Group Administration
Add Group
Edit Group
Delete Group
User Administration
Add User
Edit User
Delete User
PIN Content Types
PIN content is organized by Global Region/Country/Asset Class/Market Sector Topic Types. Chose from the following Global Regions to begin your selection process. When you have completed your selections click "Back" and press the "Submit Content Selections" button below.
Africa
The Americas
Asia-Pacific
Europe
Emerging Europe
Former Soviet Union
Middle East
For questions or suggestions, please contact xxxServices via E-mail at x.com or call 555-555-555

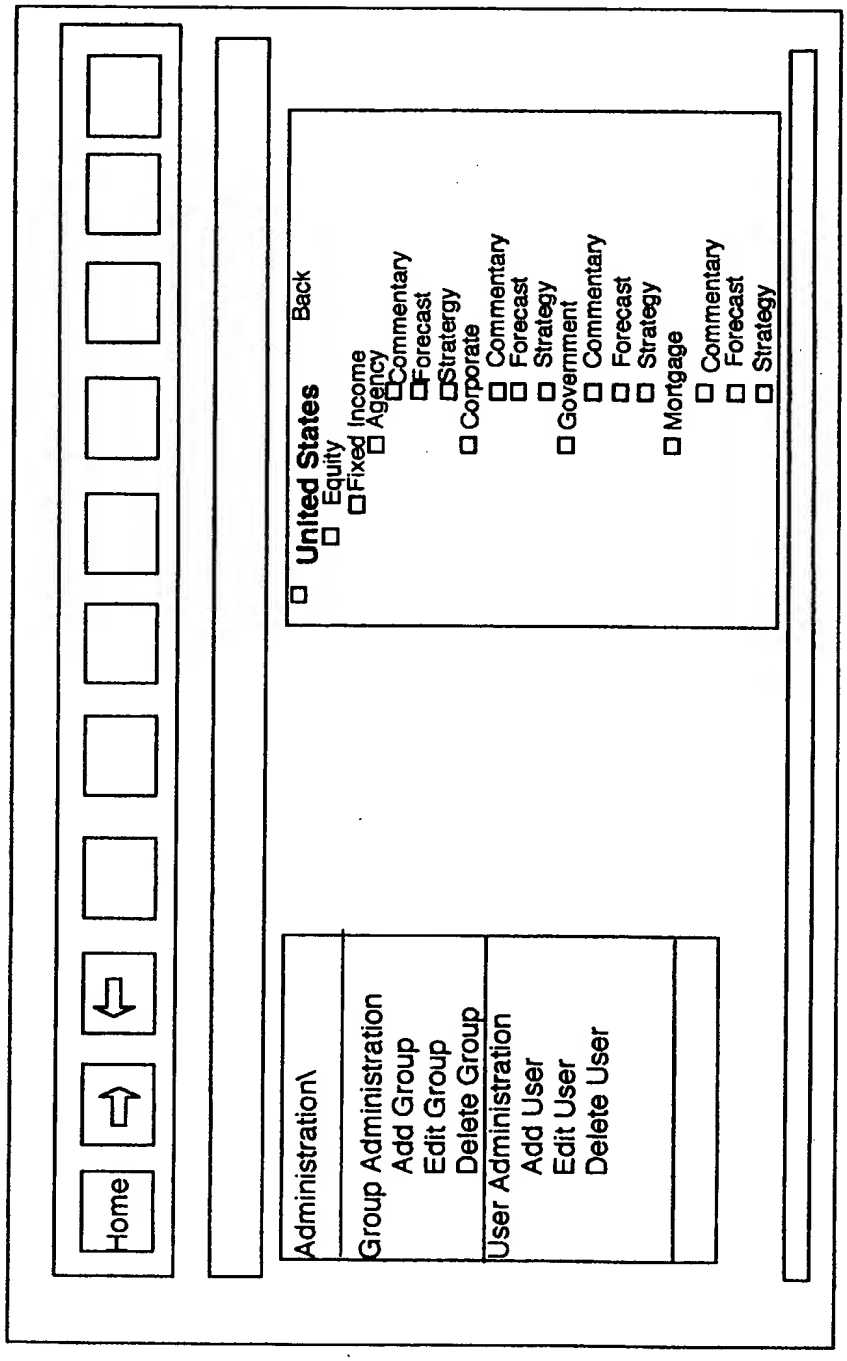

Home
Administration\
Group Administration
Add Group
Edit Group
Delete Group
User Administration
Add User
Edit User
Delete User
Back
United States
Equity
Fixed Income
Agency
Commentary
Forecast
Strategy
Corporate
Commentary
Forecast
Strategy
Government
Commentary
Forecast
Strategy
Mortgage
Commentary
Forecast
Strategy

Fig. 25b

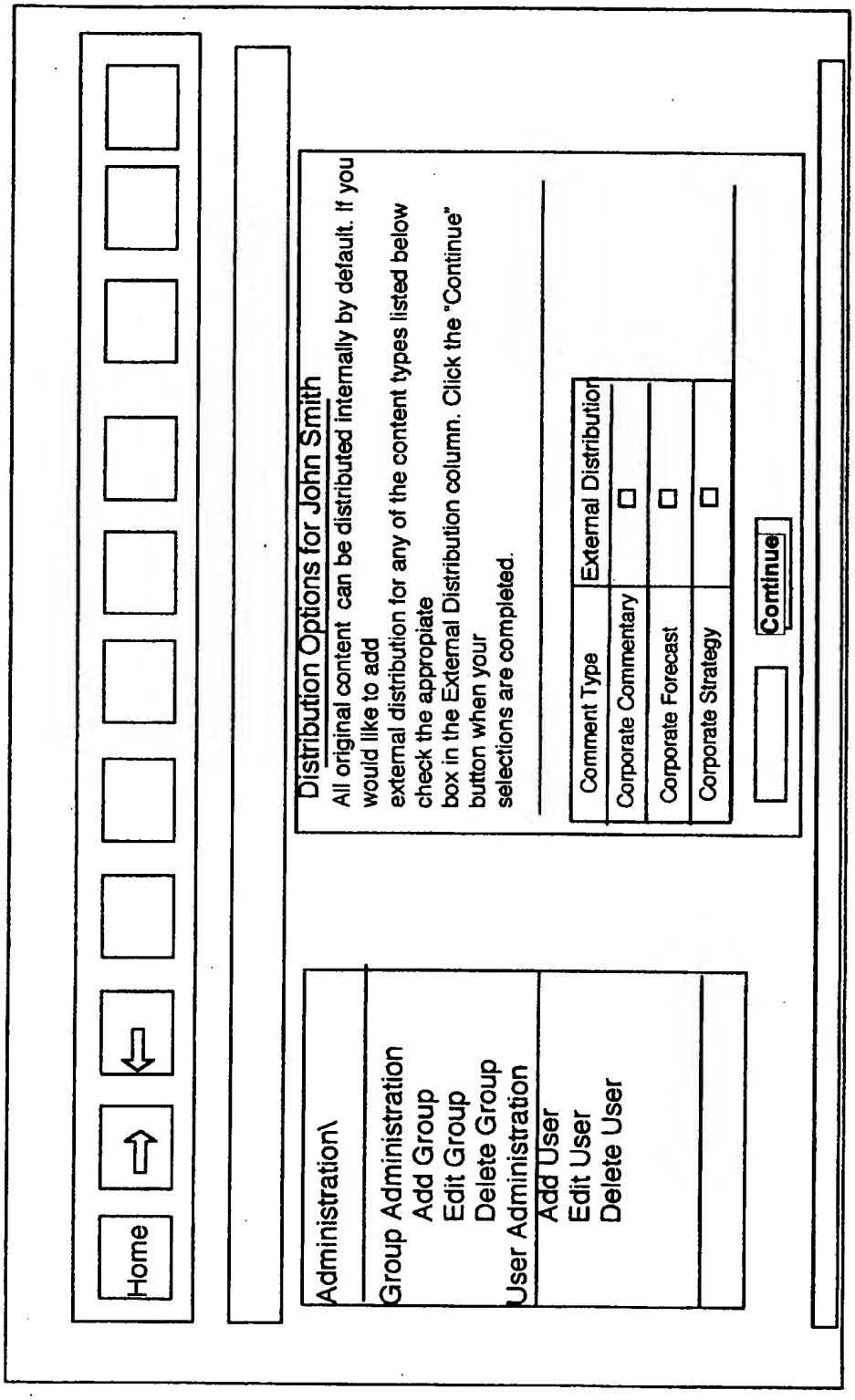

Home
Administration\
Group Administration
Add Group
Edit Group
Delete Group
User Administration
Add User
Edit User
Delete User
Distribution Options for John Smith
All original content can be distributed internally by default. If you would like to add external distribution for any of the content types listed below check the appropiate box in the External Distribution column. Click the "Continue" button when your selections are completed.
Comment Type
External Distribution
Corporate Commentary
Corporate Forecast
Corporate Strategy
Continue

Fig. 25c

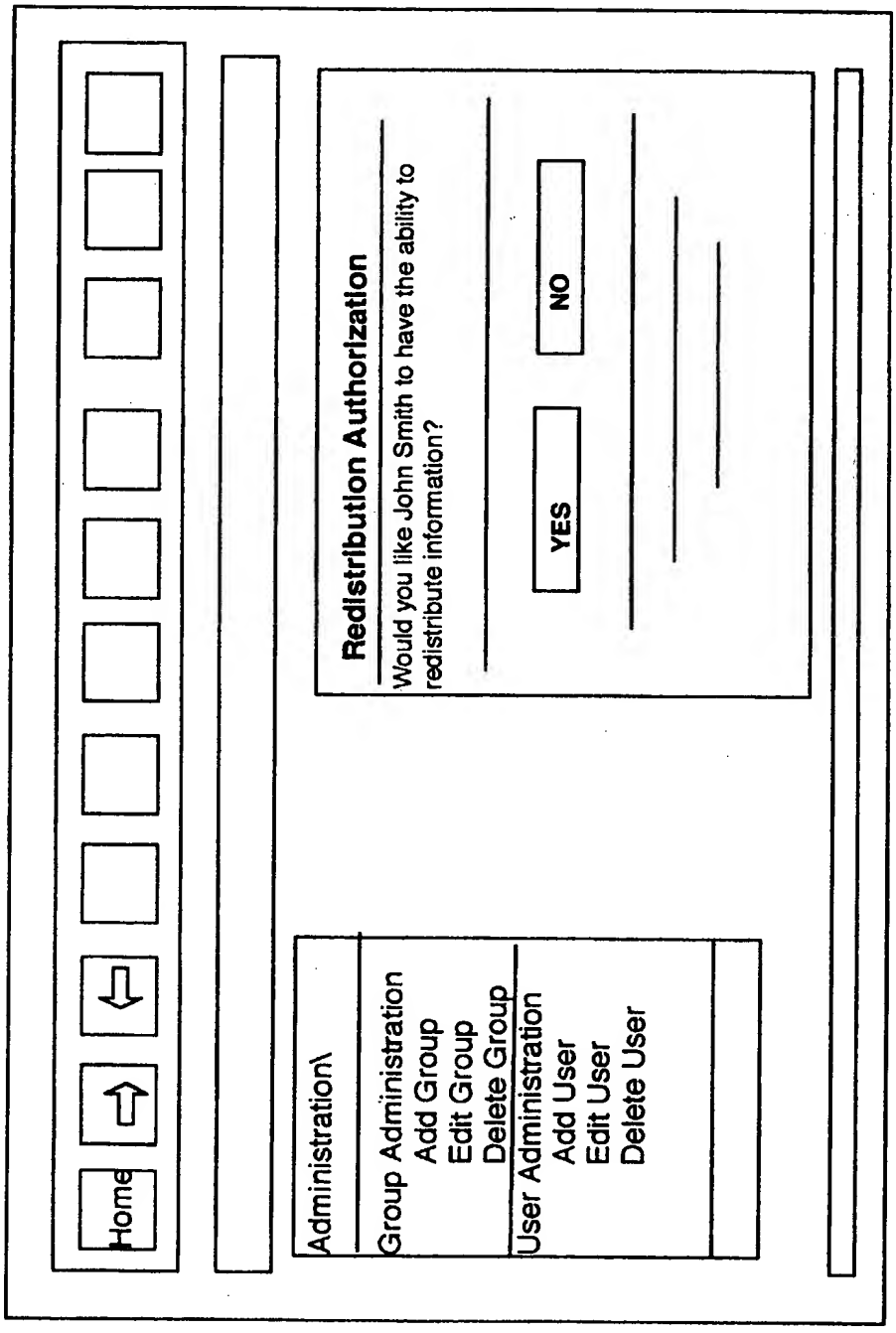

Home
Administration\
Group Administration
Add Group
Edit Group
Delete Group
User Administration
Add User
Edit User
Delete User
Redistribution Authorization
Would you like John Smith to have the ability to redistribute information?
YES
NO

Fig. 26a

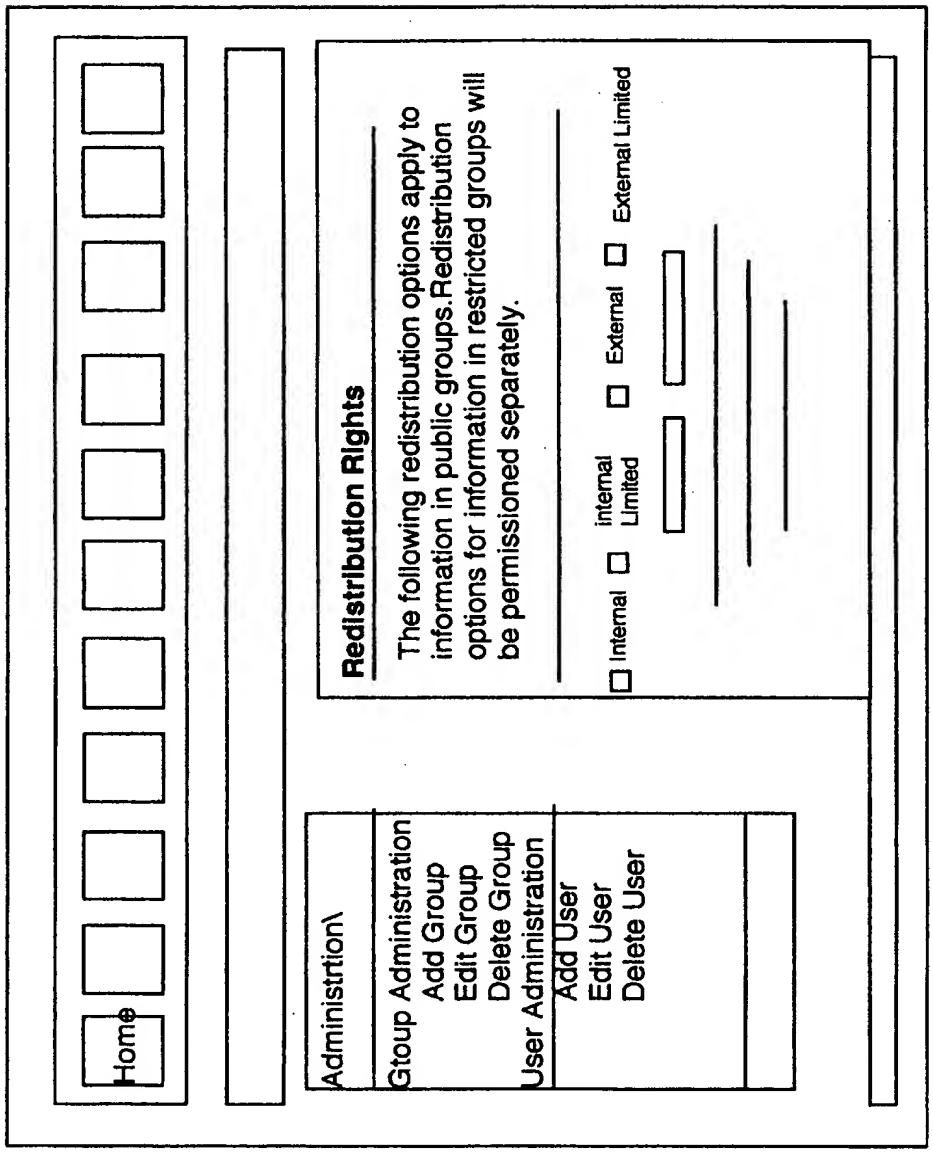

Home
Administrtion\
Gtoup Administration
Add Group
Edit Group
Delete Group
User Administration
Add User
Edit User
Delete User
Redistribution Rights
The following redistribution options apply to information in public groups.Redistribution options for information in restricted groups will be permissioned separately.
Internal
Internal Limited
External
External Limited

Fig. 26b

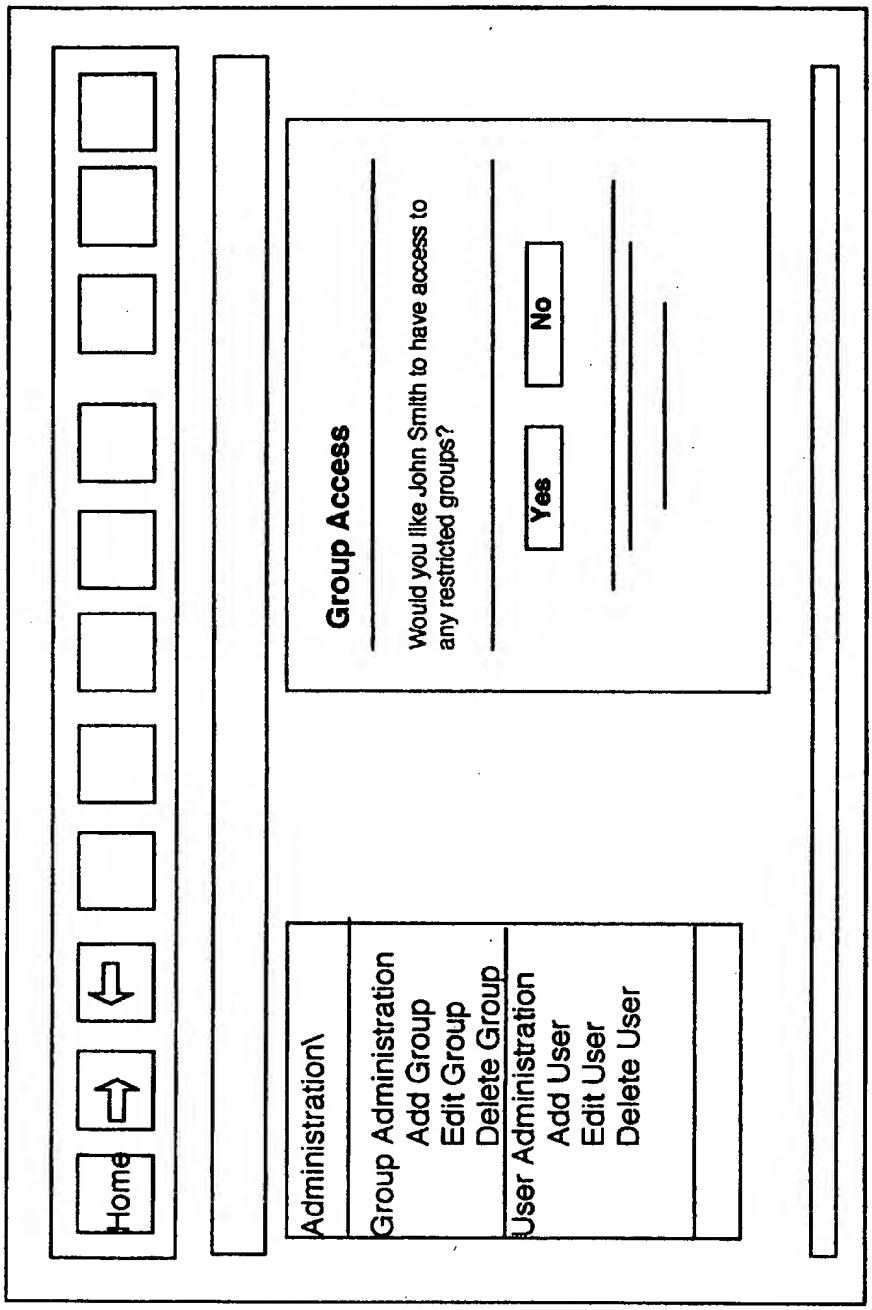

Home
Administration\
Group Administration
Add Group
Edit Group
Delete Group
User Administration
Add User
Edit User
Delete User
Group Access
Would you like John Smith to have access to any restricted groups?
Yes
No

Fig. 26c

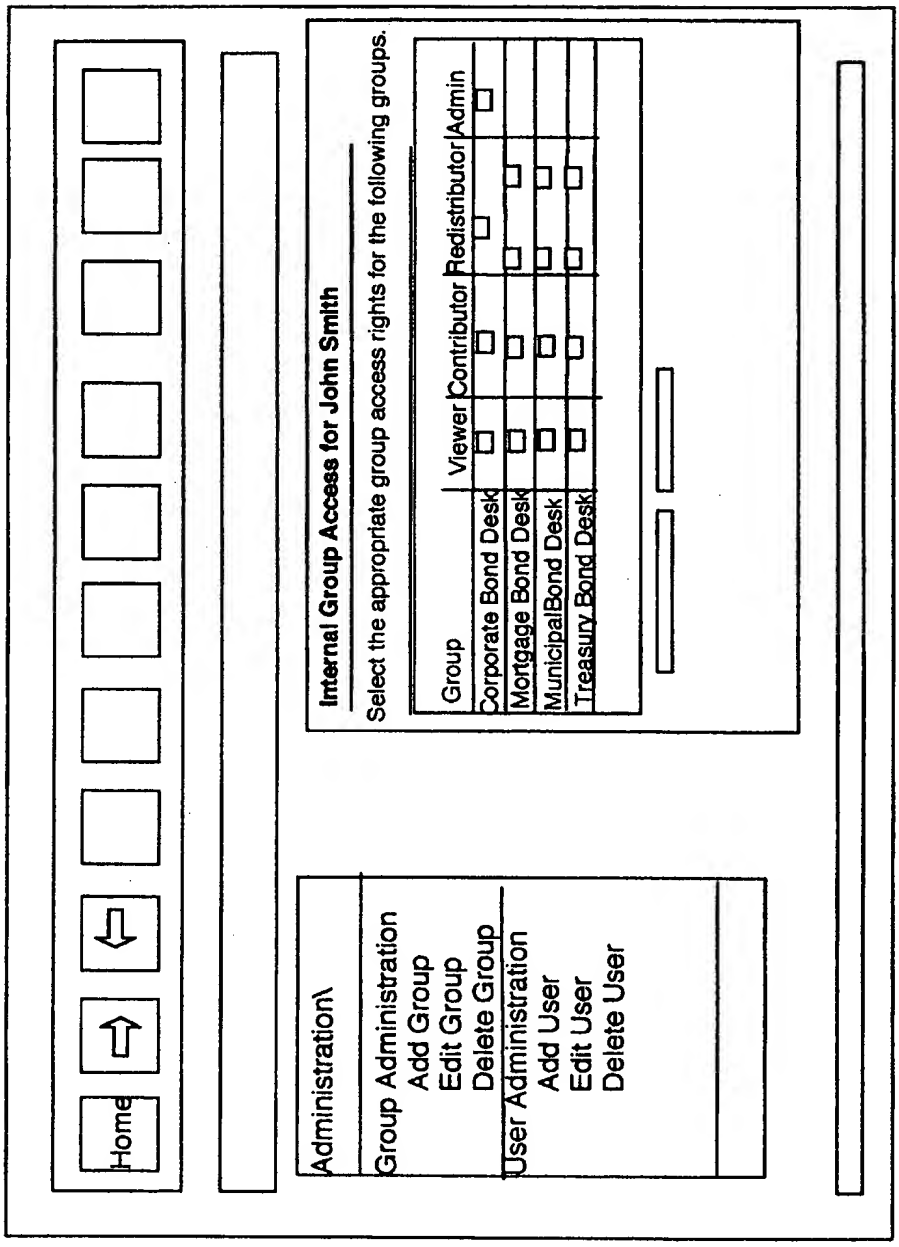

Home
Administration\
Group Administration
Add Group
Edit Group
Delete Group
User Administration
Add User
Edit User
Delete User
Internal Group Access for John Smith
Select the appropriate group access rights for the following groups.
Group
Viewer
Contributor
Redistributor
Admin
Corporate Bond Desk
Mortgage Bond Desk
MunicipalBond Desk
Treasury Bond Desk

Fig. 26d

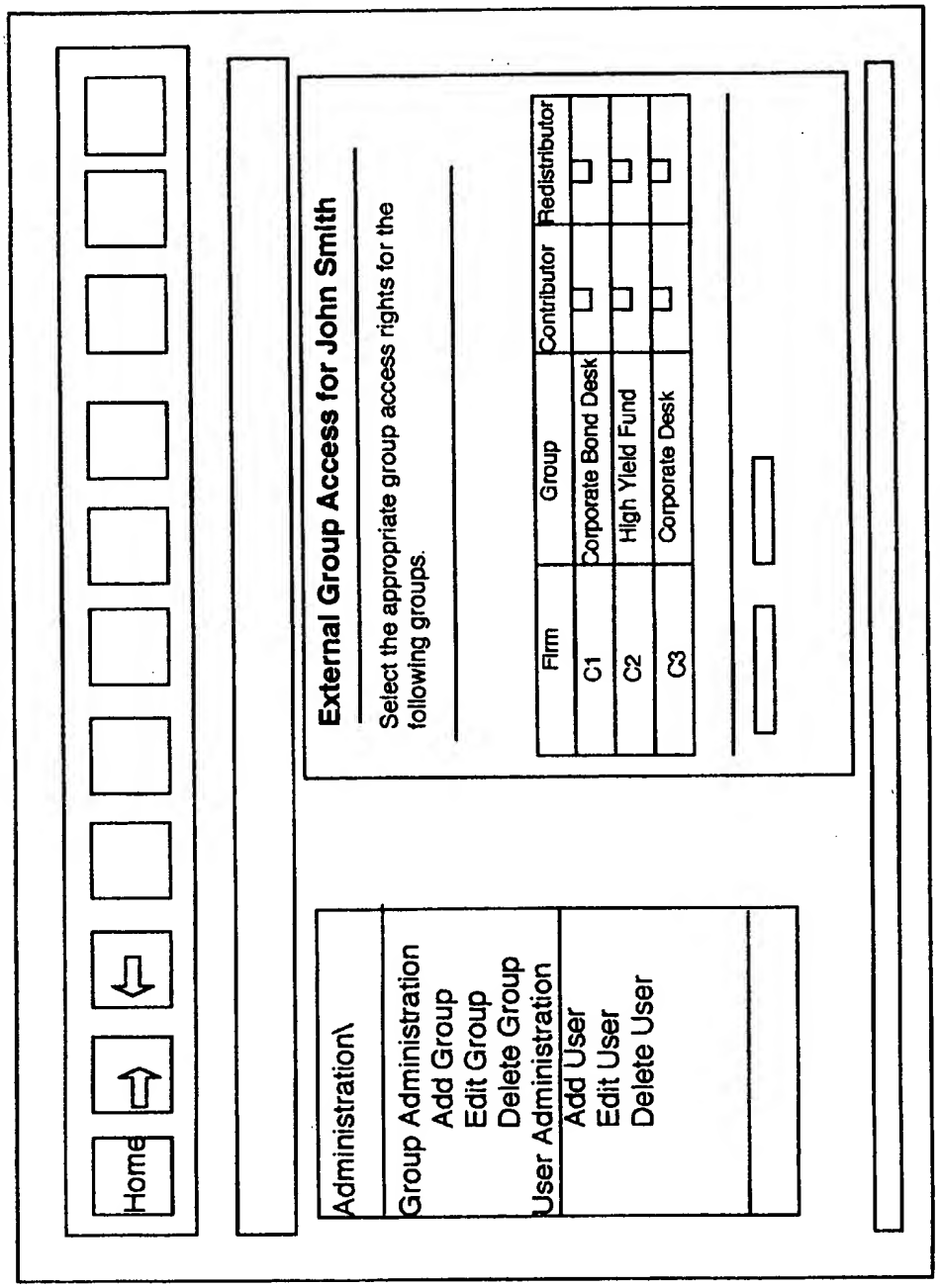

Home
Administration\
Group Administration
Add Group
Edit Group
Delete Group
User Administration
Add User
Edit User
Delete User
External Group Access for John Smith
Select the appropriate group access rights for the following groups.
Firm
Group
Contributor
Redistributor
C1
Corporate Bond Desk
C2
High Yield Fund
C3
Corporate Desk

Fig. 26e

Fig. 27

```
create table DomainClients
(DCserverName text not null,
DCactive int not null,
DCfirmID text not null,
DCfirmName text not null,
DCfirmLogo text not null,
DCsiteID text not null,
unique  (DCserverName),
unique (DCfirmID, DCsiteID));


update tables set table_owner = 'nsadmin' where table_name =
'DomainClients';
```

Fig. 28

```
create table DCClientSources
(DCSfirmID text not null,
DCSsourceFirmID text not null,
unique (DCSfirmID, DCSsourceFirmID));

update tables set table_owner = 'nsadmin' where table_name =
'DClientSources';
```

Fig. 29

```
create table DCObjects
(DCOobject large_object not null);

update tables set table_owner = 'nsadmin' where table_name =
'DDobjects';
```

Fig. 30

```
create table DCObjectDestinations
(DCODtarget text not null,
DCODIoHandle text not null,
DCODreceived timestamp not null,
DCODprocessed timestamp,
unique (DCODtarget, DCODIoHandle));

update tables set table_ower = 'nsadmin' where table_name =
'DCObjectDestinations';
```

Fig. 31

```
create table Indexes
(IndexID int not null,
IndexDescription text not null,
IndexBaseType int not null,
IndexParentID int not null,
IndexAbbrv text not null,
unique (IndexID));

update tables set table_ower = 'nsadmin' where table_name = 'Indexes';
```

Fig. 31a

```
create table Channels
(ChannelID int not null,
ChannelName text not null,
Channel Description text not null,
unique (ChannelID));

update tables set table_owner='nsadmin' where table_name='Channels';
```

Fig. 31b

```
create table ChannelContent
(CCchannelID int not null,
CCcontentID int not null,
CCtype text not null,
CCsource text not null,
unique (CCchannelID, CCcontentID));

update tables set table_owner='asadmin'where table_name='Channelcontent';
```

Fig. 31c

```
create table ChannelTemplates
(CTchannelID int not null,
CTtemplateID int not null,
CTattributes text not null,
CTsources text not null,
unique (CTchannelID, CTtemplateID));

update tables set table_owner='nsadmin'where table_name='ChannelTemplates';
```

Fig. 32a

```
create table WorkGroup
(WGgroupID text not null,
WGname text not null,
WGmembership text not null,
WGcommunity int not null,
unique (WGgroupID));

update tables set table_ower = 'nsadmin' where table_name = 'Workgroup';
```

Fig.32b

```
creat table WgroupContent
(WGCgroupID text not null,
WGCtype text not null,
WGCsourceID text not null,
unique (WGCgroupID, WGCtype, WGCsourceID));

update tables set table_owner= 'nsadmin' where table_
name='WgroupContent';
```

Fig. 33

```
create table WGroupBaseContent
(WGBCgroupID text not null,
WGBCindexID int not null,
unique (WGBCgroupID, WGBCindexID));

update tables set table_ower = 'nsadmin' where table_name =
'WGroupBaseContent';
```

Fig. 34

```
create table WGroupMasterLists
(WGMLgroupID text not null,
WGMLmasterList large_text not null,
WGMLtmeStamp timestamp,
unique (WGMLgroupID));

create index WBMLindex
on WGroupMasterLists
using btreee (WGMLgroupID);

update tables set table_ower = 'nsadmin' where table_name =
'WGroupMasterLists';
```

Fig. 35

```
create table WGroupAdhocContent
(WGACdestID text not null,
WGACsourceID text not null,
unique (WGACdestID, WGACsourceID));

update tables set table_ower = 'nsadmin' where table_name =
'WGroupAdhocContent';
```

Fig. 36

```
create table WGroupAuthorMembers
(WGAMauthorID text not null,
WGAMgroupID text not null,
WGAMrestrict int not null,
unique (WGAMauthorID, WGAMgroupID));

update tables set table_owner = 'nsadmin' where table_name =
'WGroupAuthorMembers';
```

Fig. 37

```
create table WGroupViewMembers
(WGVMgroupID text not null,
WGVMviewerID text not null,
WGVMadmin int not null,
WGVMattribute int not null,
unique (WGVMgroupID, WGVMviewerID));

update tables set table_owner = 'nsadmin' where table_name =
'WGroupViewMembers';
```

Fig. 38

```
create table AddressBook
(ABuserID text not null,
ABgroupID text not null,
ABdestination text not null,
ABattribute int not null,
unique (ABuserID, ABgroupID, ABdestination));

update tables set table_owner = 'nsadmin' where table_name = 'AddressBook';
```

Fig. 39

```
create table AuthorRights
(AuthorID text not null,
AuthorIndexID int not null,
AuthorAttribute int not null,
unique (AuthorID, AuthorIndexID));

update tables set table_owner = 'nsadmin' where table_name =
'AuthorRights';
```

Fig. 40

```
create table Viewers
(ViewerID text not null,
FullName text not null,
AdminFlag int not null,
DefWorkGrpID text not null,
DefAttribute int not null,
Passwd text not null,
unique (ViewerID));

update tables set table_owner = 'nsadmin' where table_name = 'Viewers';
```

Fig. 41

```
create table WGContentListAttributes
(WGCLAgroupID text not null,
WGCLAlistNumber int not null,
WGCLAattribKey text not null,
WGCLAattribValue text not null,
unique (WGCLAgroupID, WGCLAlistNumber, WGCLAattribKey));

update tables set table_owner = 'nsadmin' where table_name =
'WGContentListAttributes';
```

Fig. 42

```
create table WGContentListCriteria
(WGCLCgroupID text not null,
WGCLClistNumber int not null,
WGCLCcontentType text not null,
WGCLCcontentSource text not null,
unique (WGCLCgroupID, WGCLClistNumber, WGCLCcontentType,
WGCLCcontentSource));

update tables set table_ower = 'nsadmin' where table_name =
'WGContentListCriteria';
```

Fig. 43

```
create table CSAdminMessages
( CSAMlevel text not null,
CSAMclass text not null,
CSAMheading text not null,
CSAMinfo text not null,
CSAMtmeStamp timestamp not null);

create index CSAMbyOID
on CSAdminMessages
using btree(oid);

update tables set table_owner='nsadmin'where
table_name='CSAdminMessages';
```

Fig. 44

```
create table CSContent
(CSCauthorID text not null,
CSClinkOID oid,
CSCheadline text not null,
CSCfilename text,
CSCinfo large_object not null,
CSCtmeStamp timestamp not null,
CSCoriginSite text not null,
CSCoriginOID text not null,
CSCdataAttr int not null,
unique (CSCoriginSite, CSCoriginOID));

createindex CSCbyOID
on CSContent
using btree(oid);

up date tables set table_owner='nsadmin' where
table_name='CSContent';
```

Fig. 45

```
create table CSContentBase
(CSCBcontentOID oid not null,
 CSCBindexValue int not null,
 CSCBindexTree text not null,
 CSCBlocked int not null,
 unique (CSCBcontentOID, CSCBindexValue) );

create index CSCBbyContOID
on CSContentBase
using btree( CSCBcontentOID );

create index CSCBbyOID
on CSContentBase
using btree( oid);

up date tables set table_owner='nsadmin'where
table_name='CSContentBase';
```

Fig. 46

```
create table CSContentAdhoc
( CSCAcontentOID oid not null,
 CSCAdestination text not null,
 CSCAsource text not null,
 CSCAlocked int not null,
 unique ( CSCAcontentOID, CSCAdestination ) );

create index CSCAbtOID
on CSContentAdhoc
using btree(oid);

create index CSCAbyContOID
on CSContentAdhoc
using btree( CSCAcontentOID );

update tables set table_owner='nsadmin'where
table_name='CSContentIndexes';
```

Fig. 47

```
create table CSDistributionQueue
( CSDQsourceTableName text not null,
 CSDQsourceTableOID oid not null,
 CSDQsourceURL text not null,
 CSDQinsertTmeStamp timestamp not null,
CSDQretrievalTmeStamp timestamp,
 CSDQtargets text not null,
 CSDQorigins text not null,
 unique (CSDQsourceTableName,
CSDQsourceTableOID) );

create index CSDQbyOID
on CSDistributionQueue
using btree(oid);
```

Fig. 47a

```
create tableCSReplicationQueue
(CSRQsourceTableName text not null,
CSRQsourceTableOID oid not null,
CSRQsourceURL text not null,
CSRQinsertTmeStamp timestamp not null,
CSRQretrievalTmeStamp timestamp,
CSRQtargets text not null,
CSRQorigins text not null,
unique (CSRQsourceTableName, CSRQsourceTableOID));

create index CSRQbyOID
on CSReplicationQueue
using btree(oid);

update tables set table_owner='nsadmin' where
table_name='CSReplicationQueue';
```

Fig. 47b

```
create table CSReplicationQueueDestinations
(CSRQDdestId text not null,
CSRQDrepQOID oid not null,
CSRQDretrievalTmeStamp timestamp,
unique (CSRQDdestID, CSRQDrepQOID));

create index CSRQDbyOID
on CSReplicationQueueDestinations
using bytree (oid);

create index CSRQDdestIdindex
on CSReplicationQueueDestinations
using btree (CSRQDestID);

update tables set table_owner='nsadmin' where
table_name='CSReplicationQueueDestinations';
```

PUBLICATION NETWORK CONTROL SYSTEM USING DOMAIN AND CLIENT SIDE COMMUNICATIONS RESOURCE LOCATOR LISTS FOR MANAGING INFORMATION COMMUNICATIONS BETWEEN THE DOMAIN SERVER AND PUBLICATION SERVERS

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to the field of networking computer systems and more particularly to the field of systems for providing control over distribution, redistribution, access security, filtering, organizing and display of information across disparate networks.

2. Background

In most industries and professions today there is a rapidly increasing need for intercompany as well as intracompany communications. Most companies, firms, and institutions want to allow their employees to communicate internally, with other employees, and externally with the firm's customers, vendors, information sources, and others throughout a work day. Depending on the nature of the information and the relationship between the parties, these communications may need to take the form of one-to-one communiques in some cases, one-to-many broadcasts in others, many to many communications, and even many-to-one communications. Some of these categories might also provide better information for all concerned if the flow of data is interactive and collaborative, allowing recipients to comment, share, and build upon what has already been received.

At present it is both difficult and costly to achieve and manage high volumes of such communications easily, especially if extremely sensitive, confidential or proprietary information must be selectively communicated not only internally, but externally to those companies considered business partners.

In the financial industry, for example, an investment bank may want to communicate time-sensitive information to all of its investment management firm clients, and invite them to comment on it, while still insuring that the bank's competitors do not have access to the information. The investment bank may also want to receive news feeds from financial news services vendors on the same network that provides for the distribution of its proprietary information, as well as proprietary reports and analysis from other third party vendors it selects.

A decade or two ago, the tools for handling such communications would generally have been limited to telephone, facsimile, overnight mail, or, more recently, electronic mail. Each of these media had limitations and drawbacks. Overnight mail is costly and for some types of information, much too slow. The telephone is, of course, much faster, but many telephone conversations are limited to one-to-one communications, since the telephone is a synchronous form of communication requiring the partes to communicate at the same time. This is not always efficient. For an investment bank to transmit a market analysis report to its clients on a one to one basis, the process is slow and cumbersome, and inevitably some clients would get the information long before others do.

A telephone conference call insures that several clients get the same information at roughly the same time and a conference call is interactive, so that comments from various

clients can be expressed. However, if the number of people on a conference call begins to exceed some critical mass, the call may be more confusing than helpful. The voices of other clients may be mistaken for that of the investment bank's analyst, for example. In either type of voice telephone transmission of information, the recipient must take notes if he or she wants to remember details or go over the analysis later in the day. When information needs to be not only timely, but precisely and accurately recorded for later reference, voice telephonic conversation becomes less appropriate.

Modern facsimile machines permit the broadcasting of information over telephone lines to a selected group of clients, as well as the transmission of charts and graphs and other images. This also gives the clients an accurate record to refer to later. However, facsimile transmission is not interactive, so any client comments that might have been offered are lost. Recipients of facsimile transmissions usually have only a hard copy, not an electronic copy of the information, unless they use fax modems to receive. Thus, the utility to the recipient may be lowered significantly, particularly if such transmissions come into a common fax machine or mail room and take a few hours to reach the individual.

Electronic mail sent over gateways between internal corporate networks is often slow, sent in plain text format (with any visual information usually sent as an attachment, if at all), and, like faxed data, is usually not indexed. As a result, finding the information that is wanted or needed in a stream of electronic mail messages can be tedious. Recipients may also be unable to use or see the attachments unless they use the same computer software and hardware. Many companies and institutions will not allow inbound or outbound attachments to email messages for security reasons. Email technology is essentially a store and forward process that inevitably produces many copies of the same document on the same network—an inefficient use of network resources.

After encountering these problems, companies and institutions with private, internal distributed computer and telecommunication networks took another approach to addressing intercompany communications. Many gave selected customers and vendors information from the company's own internal network, by building out a separate, isolated external network to communicate with their key business partners. Selected information from the company's internal network would be sent to the special external network and then sent on to the trading partners. This allowed larger documents and files to be transferred in a secure fashion to and from external sources. However, if an institution such as an investment bank wished to do this for all of its clients and all of its vendors, expenses and complexities increased dramatically. If the investment bank used one type of computer systems and network software for its internal and external networks, and a client or vendor used another, then individuals on both sides of the communication needed to have their network administrators configure their systems to work together and develop programs to provide security, as well as functionality. This usually involved capital outlays for computers, bridges(network devices that connect two networks using two different types of media—such as 10base T cable and FDDI connections), routers (special purpose machines that connect two or more networks and route messages to the correct internet protocol address), software and terminals, plus costs for developing software to handle the connections to and from the outside. To avoid extreme costs for equipment and special development, companies tended to restrict the number of companies granted

this kind of access as well as the kind of information that could be sent or received.

To provide affordable alternatives to direct connections, other companies, such as First Call Corporation offered networking and distribution services. For example, if an investment bank wanted to deliver its research to its clients, First Call would deliver it for a fee, and also charge the recipients who received it. While this eliminated the need for intense capital expenditures and development costs on the part of both sellers and buyers of the information so distributed, it also effectively eliminated their control over the information, and its flow, too. First Call, for example, became a central source of information, not the bank or supplier, in the eyes of the clients. Since the information provided to First Call for distribution would be sent to all those who bought the service, it did not make economic sense for the providers to customize the information for any given recipient. Interactive communications were also impractical under this scheme.

Then came the Internet—the worldwide system of linked computer networks that allows thousands of existing corporate and institutional networks to communicate over it using standard communications protocols or signals. That aspect of the Internet known as the World Wide Web simplified these communications even more by providing what are known as hypertext links, and using HyperText Transport Protocol (HTTP) to allow a user to go from one hypertext link to another over the World Wide Web. (Hypertext is a way of creating and publishing text that chunks information into small units, called nodes, that have what are called hypertext links or anchors embedded in them. When a reader of the text clicks on a hyperlink, the hypertext software (also known as a browser or web browser) displays the node associated with that link. The collection of these nodes is a "web" and the Worldwide Web is a hypertext system that is global in scale.) With the Internet and the Worldwide Web, widespread dissemination of some types of information became simplified. However, most of the information published on the Internet's World Wide Web is not likely to be sensitive or confidential in nature, since access is readily available to many.

Internal corporate networks may have highly confidential business files on the same computers that form the internal network, as well as extremely confidential technical and product files that may be vulnerable to attack and theft or misuse if a connection is made between the internal network and the Internet. Consequently, most companies construct "firewalls" between their internal networks and any gateways to the external world. (See FIG. 2, where companies C1 through C9 are shown having firewalls F1 through F9, respectively.) A firewall is a security technique in which a user puts a specially programmed computer system between its internal network and the Internet. This special "firewall" computer prevents unauthorized people from gaining access to the internal network. However, it also prevents the company's internal computer users from gaining direct access to the Internet, since the access to the Internet provided by the firewall computer is usually indirect and performed by software programs known as proxy servers.

Thus, if a user wants to get a file from a vendor, he or she would send an FTP (file transfer protocol) request to the firewall computer's proxy server. The proxy server would create a second FTP request, under its name and use that one to actually ask for a file outside the network. This allows the internal names and addresses to stay inside the company. Use of firewalls and proxy servers can slow performance somewhat, and also tends to limit the types of information that can be sent or received to that which is less likely to be sensitive or proprietary.

The use of firewalls makes it less risky for internal network users to bring information in from the Internet and distribute it internally. However, once information is brought inside a private corporate network, there can still be problems distributing it internally.

Most large private networks are built of complex sets of:

Local Area Networks (LAN)-a set of computers located within a fairly small physical area, usually less than 2 miles, and linked to each other by high speed cables or other connections; and

Wide Area Networks (WAN)-groups of Local Area Networks that are linked to each other over high speed long distance communications lines or satellites that convey data quickly over long distances, forming the "backbone" of the internal network.

These private internal networks use complex hardware and software to transmit, route, and receive messages internally.

Sharing and distributing information inside a corporate network has been made somewhat easier by using client/server technology, web browsers, and hypertext technology used in the Internet, on an internal basis, as the first steps towards creating "intranets." In typical client/server technology, one computer acts as the "back end" or server to perform complex tasks for the users, while other, smaller computers or terminals are the "front-end" or "clients" that communicate with the user. In a client/server approach the client requests data from the server. A web server is a program that acts as a server function for hypertext information. In large private networks, a server computer might have web server software operating on it to handle hypertext communications within the company's internal network. At the web server site, one or more people would create documents in hypertext format and make them available at the server. In many companies, employees would have personal computers at their desks connected to the internal network. In an "intranet" these employees would use a web browser on their personal computers to see what hypertext documents are available at the web server. While this has been an advance for internal communications over a private network, it requires personnel familiar with HyperText Markup Language (HTML) the language that is used to create hypertext links in documents to create and maintain the "internal" web pages. If a more interactive approach is desired, an Information Technology (IT) specialist in some form of scripting, such as CGI, PERL, is needed who can create forms documents and procedures to allow users to ask for information from the server.

Applications that need to share information internally can also use what is known as workgroup software such as IBM's Lotus Notes™ software on the internal network. However, this, too, requires special programming and scripting for the unique needs of the organization.

It is now increasingly common for intranets to connect to the Internet forming what is sometimes called an "extranet." The Internet, however, is essentially a passive transmission system. There is no automatic notification sent to clients or customers that a new report is available on a given Internet Web page that is external to the client's intranet. Customers or clients normally would have to search the Internet periodically to see if a Web page has changed, and if the change is something he or she is interested in seeing. Some Web page sites that provide fee services use e-mail to notify prospective users that the new data is available. As mentioned, e-mail is slow, so if the data is also time-

sensitive, the notification may not reach the customer until later in the day, when it may be of much less value.

One attempt to make the Internet more interactive has been offered by Intermind, namely a form of hypertext, called hypercommunications. In this approach, a number of directories are built at various sites, in a fashion analogized to "speed dial buttons" on ordinary telephones. When a user wishes to get information from a site connected by hypercommunications, he or she "pushes" the "speed dial" button for that site, and is automatically linked to it, through directories created by the Intermind software. This approach also allows a publisher of information to poll subscribers to see if they are able to receive. If they are, and the publisher has new data to give them, the publisher "dials" his or her "speed button," thus sending the data. This helps solve the problem of notifying the customer that new information is available.

However, making information produced internally available selectively to external business partners via the Internet is an inefficient process if done manually by each author of internal information, even with such directories. Commingling internal information with external sources of information on the same intranet is also labor intensive and inefficient if done manually, even with the "speed button" approach. This approach does not provide publication control over the data, nor indexing nor organized presentation of the data. Nor does it solve the security problem posed by allowing others to access a website without a "firewall" or similar kind of access protection.

Another option that became available to an information publisher after the advent of the Internet and Web browsers was a form of connection over the Internet that provides secure access, but usually to a more limited set of information, through a "demilitarized zone" or DMZ, using encryption and secure sockets. Since each company would want to protect the privacy of the internal data on its network, each would have a firewall around its network with a "demilitarized zone" (DMZ) outside or as part of the firewall for each other company it wished to reach. As shown in FIG. 2*b*, for example, Company A's DMZ D1 might be located outside its firewall F1 between the firewall F1 and Company A's gateway G1 to the Internet. Within DMZ D1, an area IC is shown as set aside for communications to and from Company C. As can be seen in FIG. 2*b*, the DMZ's of each company that wishes to communicate directly and securely with others must be configured to identify the intended communicants.

If a customer needs to get information from 20 different external publishing sources, it may need to make 20 different connections between its firewall and that of the publishers and obtain 20 different user identifiers and passwords. A simplified illustration of this is provided in FIG. 2. For purposes of illustration, if companies C1–C3 are competing investment banks, and companies C5 through C9 are their customers, with C4 being a news source, a greatly oversimplified network configuration is shown that uses such a DMZ configuration. Notice that bank C1 has DMZ's D4–D9 for the news source C4 and the five customers C5–C9. Customer C5 has DMZ's D1–D3 for each of the investment banks it gets data from, as well as for news source C4. As FIG. 2 shows, this approach results in a maze of connections P, and DMZ's, D. A simplified view of DMZ's is shown in FIG. 2*b*, where company C1 has, in its DMZ D1, an application that communications with company C3. Company C2, has, in its DMZ D2, applications C1 and C3 to communicate with company C1 and company C3, respectively.

The DMZ approach requires each customer to obtain different user identifiers and passwords to gain access to each other company's network. For each individual at each customer site, someone in the investment bank's information technology department must assign user identifiers and passwords to each. This further requires elaborate network administration and maintenance. A setup such as this, in which the customers use Web browsers to gather information from a supplier's network, is called a "pull" model, because the customers still have to actively seek out the information. To simplify the administrative tasks as much as possible, it makes sense for the information publisher to generalize the information that goes out, so that it is sent in a one-to-many, or broadcast format. In this type of approach, one publisher may organize its information in one style, while another may structure its data quite differently. Thus, it becomes extremely difficult for the clients or customers to index or cogently organize the data from 20 different publishers.

For the information provider to be more active, a "push" model of communication is desirable. That is, rather than wait for the customers to seek out information available on its network, the provider would like to be able to notify the user that the data is there and send it out automatically. Workgroup software, such as Lotus Notes, was usually thought to be the better solution for this type of intercompany transmission. Unfortunately, this usually requires a significant amount of software development as well as administrative overhead. In the example of the customer who is getting reports and data from 20 different investment banks, the information that needs to be consolidated at an employee's desktop at the customer site usually arrives in a variety of incompatible formats. If the customer wants to get morning analyses from each bank, an information specialist at the customer site will probably have to find out what format is used by each sending bank, have the customer's programmers understand the network address schemes, as well as the protocols, packets, ports and sockets to be used for each bank, and then create or modify one or more Lotus Notes workgroup application programs at the customer's employee's desktop to convert the data into an internal format and bring it in.

One attempt to address at least part of these problems is a technique known as "subject-based addressing technology" as described in U.S. Pat. No. 5,557,798 assigned to Tibco. Using this approach, and the example of the direct network to network connection via a DMZ, shown in FIG. 2*a*, a publisher C1 might set up a server at its site to publish information by subject. The customer C5, usually has a "client" application, in its DMZ D5. The client application denotes the set of messages to receive using human-readable subject names. Subject-based addressing can eliminate the need for the customer programmers to understand all the network address, protocol, packet, port and socket details, and even simplifies some of the modification that needs to be done to the workgroup software. However, it does not eliminate the need to configure conversion or translation layer software at the site to take a network feed, and to understand how the data that is transmitted is formatted, and the need to modify the workgroup software, such as Lotus Notes applications, accordingly. In fact, both subject-based addressing and workgroup software such as Lotus Notes usually require a significant amount of additional programming development work to be done by the users in order to work effectively.

From the information publisher's perspective, a "push" model that relies on the private network-to-network connec-

tion through firewalls, DMZ's and workgroup programs, and uses subject-based addressing still fails to address the distribution control problem that may be vital to the publisher. If the investment bank C1 of FIG. 2*a* provides a morning analysis as a subject, once the data crosses out of the bank's network and is disseminated over the Internet, the investment bank has usually lost all control of replication of the analysis. In most cases of subject-based addressing, the publisher will not even know which companies are consuming its information.

Even if one set of programs is written to address publication control and dissemination at one customer site, such as customer C8, (in FIG. 2*a*) for example, using either software such as Lotus Notes or subject-based addressing, it is not always simple or easy to adapt that set of programs to work with customer C9's network, or amongst several different customer's networks. Once it becomes desirable or necessary to send and receive information over the Internet or a wide area network linking several different corporations, dissemination control becomes a very complicated problem.

As already mentioned, it is difficult to index or organize information received from many different sources so that it can be grouped the same way on every receiver's desktop. Some profiling or "filtering" systems (such as products from Individual or Pointcast) gather data from public sources and filter or sift through them to select information tailored to an individual person's request, but these systems do not usually control replication, nor do they allow any interaction with the data. Profilers are usually one-to-many, one way distribution models that do not allow any interaction.

In corporations and large institutions with intranets, where browsers are used, individual receivers of information can organize what they see by keeping bookmarks. However, bookmarks are usually so customized that no two sets of them are likely to be identical. As with the external profiling systems, intranets using browsers and bookmarks are also usually only able to send information in one direction. A user at company C8 of FIG. 2 who gets the analysis provided by bank C1, usually cannot use a browser to comment and reply, unless a special form sheet has been created by using CGI scripting or some other programming or scripting language for that purpose for that Web page, by bank C1. Again, custom programming or scripting adds to costs and usually makes it difficult to standardize across companies.

Most intranet systems connected to the Internet today do not allow an individual user to request information by both source and subject, and most do not allow an individual user to act as both an author and a viewer of information.

As FIG. 2*a* illustrates, connecting consumers of information over the Internet to external information sources via DMZ's and secure sockets is complex and cumbersome, as well as costly to set up and administer for the publishers of information. From the viewpoint of the consumers of information over the Internet it should be noted that transmissions over such a distribution model occur at "Internet speed." That is to say, once a request for information leaves customer C8, for example, if it goes over the Internet it is in TCP-IP formatted packets, and possibly encrypted via secure socket technology. In any case, its speed is the average speed of the Internet transmission links, once it leaves customer C8's backbone network. This is usually much slower than the speed of transmission within the customer's own internal network. Thus, performance speed of the intercompany communications can be problematic as well, when seen from the consumer's viewpoint.

While the use of DMZ's or devices such as proxy servers help ameliorate the security problems, DMZ's also tend to create content backlogs that form bottlenecks for all intercompany communications. For example, if the only persons authorized to transfer data outside the company's firewall to its DMZ are the information technology specialists, this can become a labor intensive chore or a bottleneck or both for a company that needs or wants to send a high volume of information outside selectively. Similarly, present security technology provides various encryption options (thus creating problems for standardization amongst companies) but leaves such matters as identification up to the information technology (IT) department at each company to manage. The IT specialists must assign user identifiers and passwords to every external individual authorized to access information (authentication) in the company's DMZ. Presently this is usually done by manual letters of reference and manual data entry of each business and individual.

If, as mentioned, documents must be created using HTML, or special CGI (common gateway interface) scripts also need to be created and maintained to put data into the proper formats, all of this tends to place matters of policy and content management in the hands of IT department specialists, rather than in the hands of authors and viewers of information. IT specialists within companies are being overwhelmed by requests to add new users and individuals, administer the types of data that can be transmitted and create maintain changes and updates to the scripts, programs, networks and systems as a whole.

It is an object of this invention to provide a universal domain routing and publication control system that enables the selective transmission of valuable information in a manner that allows for control of replication and publication of the information.

It is another object of the invention to provide a system that can disseminate information selectively between disparate types of users and networks.

Still another object of the present invention is providing a system that allows users to comment on and interact with the information received.

It is another object of the present invention to minimize or eliminate the need for software development by users and information providers.

Another object of the present invention is reducing the need for special administrative procedures and specially trained personnel to manage the system.

Still another object of the present invention is providing a system that allows users to access information at the speeds of their internal networks the majority of the time.

Another object of the present invention is providing dynamic distributed network resource registries that facilitates the standardization and organization of information by subject, source or a combination of both.

SUMMARY OF THE INVENTION

These and other objects are achieved by a publication control system for networks inside the same client entity having several publication computers in communications relationship with each other inside the client entity, each of said publication computers having disks for storing a dynamic group registry and resource locators containing function names, each also having a web server program which, when executed by the publication computer, causes the publication computer to respond to resource locators by calling the function name indicated, each publication computer also has a database management program for organizing the dynamic group registry; and each publications com-

puter has a client side communications server program, which, responds to resource locators directed to the client side communications server program and directs the database management program in organizing the dynamic group registry; the system also has a domain computer having a disk for storing a dynamic client registry and resource locators containing function names; a web server program which, when executed by the domain computer, causes the domain computer to respond to the resource locators by calling the function name indicated, the domain computer also having a database management program for organizing the dynamic client registry; a domain communications server program which, when loaded by the web server program responding to the appropriate resource locator, is executed by the domain computer, to respond to resource locators directed to the domain communications server program and to direct the database management program in organizing the dynamic client registry; a domain communications resource locator list stored in the domain computer and each publication computer that causes predetermined functions to be executed in the domain communications server; a client side communications resource locator list stored in the domain computer and each publication computer that causes predetermined functions to be executed in the client side communications server in each publication computer so that communications between the domain computer and each publication computer cause the selected functions to be executed dynamically in order to manage information communications between the domain computer and each publication computer.

It is an aspect of this invention that it allows a company to communicate securely with other firms outside its private network without requiring the use of DMZ's at any site.

It is an aspect of this invention that it provides a dynamically configurable domain routing and publication control system.

Another aspect of the present invention is that it enables users to define and implement their own policies for distribution and redistribution of information on a network.

Still another aspect of the present invention is that it does not require additional software development by users.

Yet another aspect of the present invention is that it does not require additional skilled information technology personnel to administer the system, but instead, allows users to administer it themselves.

Another aspect of the present invention is that it makes it possible for users in a private network to receive information from outside the network at the speed of the private network most of the time.

Yet another aspect of the present invention is that it gives information publishers a simple way to produce selective distributions to various clients.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a schematic diagram of the present invention showing a domain communications server and several client side communications servers.

FIG. **1***a* is an alternative schematic diagram of the present invention showing virtual connections between a domain communications server and several client side communications servers.

FIG. **2***a* is a schematic diagram of private networks communicating externally over the Internet using prior art.

FIG. **2***b* is a schematic diagram of private networks communication externally over the Internet using prior art.

FIG. **3** is a schematic diagram of the present invention showing several domains in communication with each other.

FIG. **4** is a schematic diagram of a client side communications server according to the method and apparatus of the present invention.

FIG. **5** is a schematic diagram showing illustrative interconnections between a domain communications server and several client side communications servers according to the method and apparatus of the present invention.

FIG. **6***a* is a block diagram of an illustrative dynamic client registry at a domain communications server according to the method and apparatus of the present invention.

FIG. **6***b* is a detailed block diagram illustrative of the fields of a dynamic client registry at a domain communications server of the present invention.

FIG. **7***a* is a block diagram of an illustrative dynamic group registry at a client side communications server according to the method and apparatus of the present invention.

FIG. **7***b* is a detailed block diagram illustrative of the fields of a dynamic group registry at a client side communications server of the present invention.

FIG. **8***a* is a list of principal domain communications server uniform resource locators (URL's) used in a preferred embodiment of the present invention.

FIG. **8***b* is a list of principal client side communications server URL's used in a preferred embodiment of the present invention.

FIG. **9** is a detailed layout of an illustrative domain clients list according to the method and apparatus of the present invention.

FIG. **10** is a detailed layout of an illustrative client sources list of the present invention.

FIG. **11** is a detailed layout of an illustrative client objects list of the present invention.

FIG. **12** is a detailed layout of an illustrative object destinations list of the present invention.

FIG. **13** is a detailed layout of illustrative indexes according to the method and apparatus of the present invention.

FIG. **14** is a flow diagram of startup procedures in a client side communications server according to the method and apparatus of the present invention.

FIG. **15** is a flow diagram of initialization of shared objects by a client side communications server according to the method and apparatus of the present invention.

FIG. **16** is a more detailed flow diagram of initialization of shared objects by the client side communications server according to the method and apparatus of the present invention.

FIG. **17** is a flow diagram showing shared object initialization by the admin server of the client side communications server according to the method and apparatus of the present invention.

FIG. **18** is a flow diagram showing shared object initialization by the tool server of the client side communications server according to the method and apparatus of the present invention.

FIG. **19** is a flow diagram showing shared object initialization by the agent server of the client side communications server according to the method and apparatus of the present invention.

FIG. **20***a* is a flow diagram showing the initialization of the dynamic client registry by the domain communications server according to the method and apparatus of the present invention.

FIG. 20*b* is a flow diagram showing the creation of a group by the client side communications server according to the method and apparatus of the present invention.

FIG. 21 is a flow diagram showing the initialization of a client side communication server according to the method and apparatus of the present invention.

FIG. 22 is a block diagram of an illustrative user screen display at a desktop terminal for adding a user according to the method and apparatus of the present invention.

FIG. 23 is a block diagram of an illustrative user screen display for entering a new user's account information according to the method and apparatus of the present invention.

FIG. 24 is a block diagram of an illustrative user screen display for authorizing content creation for a new user according to the method and apparatus of the present invention.

FIG. 25*a* is a block diagram of an illustrative user screen display for selecting content types according to the method and apparatus of the present invention.

FIG. 25*b* is a block diagram of an illustrative user screen display for selecting content type according to the method and apparatus of the present invention.

FIG. 25*c* is a block diagram of an illustrative user screen display for selecting distribution options for a user according to the method and apparatus of the present invention.

FIG. 26*a* is a block diagram of an illustrative user screen display for authorizing redistribution rights generally according to the method and apparatus of the present invention.

FIG. 26*b* is a block diagram of an illustrative user screen display for authorizing redistribution rights more specifically according to the method and apparatus of the present invention.

FIG. 26*c* is a block diagram of an illustrative user screen display for assigning group access according to the method and apparatus of the present invention.

FIG. 26*d* is a block diagram of an illustrative user screen display for assigning internal group access for a user according to the method and apparatus of the present invention.

FIG. 26*e* is a block diagram of an illustrative user screen display for assigning external group access for a user according to the method and apparatus of the present invention.

FIG. 27 is pseudo-code in SQL format illustrating the creation of a domain clients table.

FIG. 28 is pseudo-code in SQL format illustrating the creation of a client sources table.

FIG. 29 is pseudo-code in SQL format illustrating the creation of a client objects table.

FIG. 30 is pseudo-code in SQL format illustrating the creation of an object destinations table.

FIG. 31 is pseudo-code in SQL format illustrating the creation of an index table.

FIG. 31*a* is pseudo-code in SQL format illustrating the creation of a channels table.

FIG. 31*b* is pseudo-code in SQL format illustrating the creation of a channel content table.

FIG. 31*c* is pseudo-code in SQL format illustrating the creation of a channel templates table.

FIG. 32*a* is pseudo-code in SQL format illustrating the creation of a workgroup table **70**.

FIG. 32*b* is pseudo-code in SQL format illustrating the creation of a workgroup content table

FIG. 33 is pseudo-code in SQL format illustrating the creation of a workgroup base content table.

FIG. 34 is pseudo-code in SQL format illustrating the creation of a workgroup master list table.

FIG. 35 is pseudo-code in SQL format illustrating the creation of a workgroup ad hoc content table.

FIG. 36 is pseudo-code in SQL format illustrating the creation of a workgroup author members table.

FIG. 37 is pseudo-code in SQL format illustrating the creation of a workgroup view members table.

FIG. 38 is pseudo-code in SQL format illustrating the creation of and address book table.

FIG. 39 is pseudo-code in SQL format illustrating the creation of an author rights table.

FIG. 40 is pseudo-code in SQL format illustrating the creation of a viewers table

FIG. 41 is pseudo-code in SQL format illustrating the creation of a workgroup content list attributes table.

FIG. 42 is pseudo-code in SQL format illustrating the creation of a workgroup content list criteria table.

FIG. 43 is pseudo-code in SQL format illustrating the creation of a client server administrative messages table.

FIG. 44 is pseudo-code in SQL format illustrating the creation of a client server content table.

FIG. 45 is pseudo-code in SQL format illustrating the creation of a client server content base table.

FIG. 46 is pseudo-code in SQL format illustrating the creation of a client server ad hoc content table.

FIG. 47 is pseudo-code in SQL format illustrating the creation of a client server distribution queue table.

FIG. 47*a* is pseudo-code in SQL format illustrating the creation of a client server replication queue table.

FIG. 47*b* is pseudo-code in SQL format illustrating the creation of a client server replication destinations queue table.

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 shows a preferred embodiment of the present invention in a schematic block diagram. A domain communications server A1, is in communication with a number of client side communications servers C1 through C9. Each of these client side communications servers is located behind the firewall F, of its respective corporate site in a preferred embodiment. That is, assume Company C-One has a client side communications server C1, Company C-Two has a client side communications server C2, and so on, through Company C-Nine. Temporary logical connections or "pipes" P are made between each client side communications server in this domain and the domain communications server A1. In a preferred embodiment, pipes P are connections formed over the Internet using conventional TCP-IP networking protocol and Netscape Corporation's secure sockets with encryption technology. However, as will be apparent to those skilled in the art, other networks and protocols and encryption techniques could be used as well.

Still in FIG. 1, it can be seen that each client side communications server C has only one actual communications pipe P with domain communications server A1. Yet, depending on the way in which each client is registered with domain communications server A1, information may be disseminated from client side communications server C1 to any or all of the other client side communications servers C2 through C9. Thus, the present invention creates an intelli-

gent extranet that links the community of companies C-One through C-Nine together over a network by means of domain communications server A1.

The intelligent extranet that is created by the present invention allows each member client C to communicate with those companies or institutions external to it (which have authorized communicating with C) as though the other company were a part of the client's own internal network or intranet. In this sense, the invention creates a virtual community of corporate intranets.

To use a more specific example where the community of companies is in the financial industry, if C1, C2 and C3 are client side communications servers for three different investment banks, and C5 through C9 are client side communications servers for investment management firms, while C4 is the client side communications server for a news broadcast organization, the investment bank at client side communication server C1 is able, by communicating directly only with domain server A1, to send information to any of the others in communication with domain communications server A1. That is, if it wishes, the bank C-One at client side communications server C1 can send its morning analysis reports only to client side communications servers C5 through C9, at customers C-Five through C-Nine, while the news broadcast organization, C-Four, may broadcast its information to all the other client side communications servers communicating with domain communications server A1. Thus, a preferred embodiment of the present invention also works as a relationship information manager for the participating companies or entities in this community. The present invention thus allows the customers of investment bank C-One to receive valuable information in a timely way from a trusted advisor over a secure connection.

Turning to FIG. 1*a*, some of the types of communications possible are illustrated. In this example, 4 different clients are shown, C1 through C4. Client C1 might be an investment bank located in Pennsylvania, USA, with only one client side communications server C1-PA acting as its "smart intranet," according to the method and apparatus of the present invention. Client side communications server C1-PA is connected to the bank's internal Local Area Network C1-Lan, which also includes two terminals, T1 and T2.

As will be apparent to those skilled in the art, a terminal T could be any type of device capable of connecting to a network, such as a computer, a mini-computer, a workstation, a personal computer, a CRT terminal with keyboard, an internet-equipped television terminal, a keyboard or touch screen and display device, a personal digital assistant, or any device that allows a user to communicate with a network and see information displayed. In a preferred embodiment, a personal computer capable of being connected to a network is used, together with standard Web browser software executing in the personal computer.

In a preferred embodiment, the term "smart intranet" describes the way the present invention enables a company's internal network—its "intranet"—to perform information management (production, access and replication of information), on a one to one, group to group and site to site basis both inside the company as well as with participating companies external to it.

Still in FIG. 1*a*, client C2 might be an investment bank C2 that has offices in Hong Kong, New York and London, all connected with each other through the bank's wide area network C2WAN to form an internal network. The bank's entire network is shielded from external intrusion by firewall F2. Each of investment bank C2's sites at Hong Kong, New York and London has its own Local Area Network—C2-HKLan in Hong Kong, C2- NYLan in New York, and C2-LNLan in London, with terminals T using standard commercially available Web browsers also connected at each Local Area Network.

As shown in FIG. 1*a*, each site of client C2 has a client side communications server (C2-HK, C2-NY, C2-LN) connected to its Local Area Network, and thus to client C2's Wide Area Network, C2WAN, forming for client C2 a smart intranet, according to the method and apparatus of the present invention.

Similarly, client C4 has multiple sites connected by a Wide Area Network behind a firewall to form its smart intranet, while client C3 has only one site, its Boston location, in communication with domain communications server A1 as its smart intranet.

Still in FIG. 1*a*, it can be seen that logical or virtual flow of smart intranet communications at client C2 are shown by the dotted line "pipes" P2 connecting each LAN inside client C2. The term pipes is used here to denote a logical connection managed through software, not necessarily a physical one created by hardware. As will be apparent to those skilled in the art, there are many ways in which the Local Area Networks within a client can be physically connected to each other over a Wide Area Network to form an internal network.

In a preferred embodiment of the present invention, as shown in FIG. 1*a* these logical internal connecting pipes P2 also exist outside client C2, and extend to domain communications server A1, and through it, to clients C1 and C3, but not to client C4. That is to say, in a preferred embodiment of the present invention, a client such as client C2 can communicate logically (through the present invention) with designated external sites as though they were internal to it or part of its own smart intranet and vice-versa. Yet Client C2 has not lost any of the protections afforded by its firewall F2, since client C2 makes no physical connection with any of these external sites except domain communications server A1. In a preferred embodiment, all physical transmissions between client C2 and domain communications server A1 take place over the Internet using Netscape Corporation's Secure Socket Layer (SSL) technologies and encryption.

Still in FIG. 1*a*, it can be seen that each client C, has its own "pipe " P, ranging from P1 through P6, as indicated in the legend 00. For example, the three sites at client C2 communicate with each other internally over "pipe" P2. In a preferred embodiment of the present invention, client C2 also seems to communicate over pipe P2 with clients C1, C3, and C5 while, in fact, client C2 has only one actual pipe connecting it to domain communications server A1. As will be apparent to those skilled in the art, this single pipe between client C2's network and domain communications server A1 could also be implemented as a direct physical connection, if desired, using T1 or T3 high speed lines. The present invention allows client C2 to communicate externally with clients C1, C3 and C5 through domain communications server A1 over a set of "virtual pipes" VP2. The term virtual pipe is used here to indicate that communication occurs as it would if two (or more) clients were in direct communication with each other over the Internet (or other network), when, in fact, each client is communicating physically only with a domain communications server and is only in "virtual" communication with the other clients or firms. The virtual pipes are created and managed dynamically by the logic of the present invention's domain communications server working with the client side communications servers at each site. Hence the term, intelligent extranet.

Still in FIG. 1*a*, note that domain communications server A1 acts as the primary domain communications server for the clients shown, but is also connected to regional or alternate domain communications servers A2 and A3. Consequently, if the computer systems or the physical communication links to the primary domain communications server A1 should fail for any reason, in a preferred embodiment of the present invention, communications can continue through use of an alternate domain communications server A2 or A3. In a preferred embodiment of the present invention, domain servers keep each other up to date by automatic replication of all relevant tables and data, thus acting as mirrors to each other.

As will be apparent to those skilled in the art, different domains could also be linked to each other by having a hyper or super domain communications server that maps a community of domain communications servers together. Similarly, the domain routing of the present invention can also be used internally to create additional domain communications servers to offload workloads from each other in large internal networks. For example, companies having client side communications servers at several locations in the US and in Europe, might choose to have the European client side communications servers connect to a domain communications server in Europe which is mapped into a corporate domain communications server in the US, which, in turn, might be mapped into external domain communications servers as described above. Using domain communications servers internally to offload work can improve the response time and throughput throughout the network.

Turning now to FIG. **3**, multiple connections of domain communications servers are shown. Here domain communications server A1, which may be acting as the primary domain communications server for a network of clients, can also be an alternate domain communications server for the networks controlled by domain communications servers A2 or A3 or both. Alternatively, domain communications servers A2 and A3 might be regional domain communications servers for a single network controlled by domain communications server A1.

With reference now to FIG. **4**, a preferred embodiment of the present invention is depicted schematically. As shown in FIG. **4**, at a client C8's Boston location, C8-BO, a computer **20** is shown connected to domain communications server A1 from behind firewall F8, and also to client C8's Local Area Network C8Lan. In a preferred embodiment of the present invention, America Online Corporation's standard Webserver software WS, known as AOLserver TM is installed at computer **20** to handle TCP-IP communication protocols between computer **20**, firewall F8, and the Internet as well as client requests from the browsers BR on the terminals T1–T4.

As will be apparent to those skilled in the art, any Webserver software or similar program that handles general communications protocols and transport layer activities could be used as appropriate for the network protocol in use. Similarly, database management software DBMS, DBMS8 in this example, is used to store and maintain a client side dynamic group registry **07** located on local electronic storage media such as disk **08** connected to computer **20**. In a preferred embodiment, the Illustra™ object-oriented relational database software supplied by Informix Corporation is used, since this allows the use of object-oriented relational technology. However, as will be apparent to those skilled in the art, any commercially available database management software could be used to store and maintain data in client side dynamic group registry **07** stored on disk **08**. Similarly, any of a number of electronic storage media could be used instead of disks. For example, in computers having sufficient random access memory, internal memory could be used as the electronic storage media.

Still in FIG. **4**, a client side communications server CSS, here CSS8, is shown executing on computer **20** at client C8's Boston site. In a preferred embodiment, a client side communications server CSS is used at each customer or client's site to serve all content produced internally, by the client, and also to handle reception of all content distributed from outside the client but within the domain served by domain communications server A1. A client side communications server for a given customer may be made up of more than one server executing on other computers **20**, but, in a preferred embodiment, each server for that given client uses replication to insure that the appropriate authorized information at each site is the same. Note that each customer is not necessarily authorized to have access to all the content in the domain. In fact, the present invention is specifically designed so that access to content throughout the domain can be directed and controlled. Also in a preferred embodiment a client side communications server CSS for a given customer must use a domain communications server to communicate with other customers that are external to it.

Referring now to FIG. **5**, a schematic diagram of several clients in communication with a domain communications server A1 is shown. As can be seen, client C8 has the address of domain communications server A1 on local disk **08** coupled to client C8's computer **20**, in which client side communications server software CSS8 is executing. Note also in FIG. **5** that domain communications server A1 includes a computer **20**, and a local storage media, in this case, disk **40**, having a dynamic client registry **06** stored in it. Also at domain communications server A1 it can be seen that a conventional Webserver WS is executing in computer **20**, together with a conventional database management program DBMS. In a preferred embodiment of the present invention, domain communications server software DSS also executes in cooperation with Webserver WS and database software DBMS. It can be seen that domain communications server software DSS maintains a list of all clients, C1–C8, in this case, that are part of this domain in dynamic client registry **06** on disk **40**.

In a preferred embodiment, web server WS is the AOLserver product from America Online, as it also allows the use of object-oriented technology. In open systems, such as Unix and similar operating systems, shared objects can be created in the C++ language. In the C++ programming language, a shared object is simply compiled code. AOLServer has the ability to dynamically initialize shared objects from URL's registered with the web server WS, so that those shared objects have callbacks. Which shared object to load is specified within an initialization file used when starting AOLServer's NSD process (the executable form of the AOLserver.)

Similarly, the DBMS software used by domain communications server A1 in a preferred embodiment is the Illustra software mentioned above. Also in preferred embodiments, computers used as either domain communications servers or client side communications servers can be any of a number of commercially available types, from mainframes to minicomputers to workstations or personal computers. Preferred embodiments of the present invention are designed to work with a number of existing installations as "middleware"—meaning software that does not replace or substitute for the existing operating system software or the typical basic communications software. Nor is it a substitute for applications software, such as the web browser. Instead, it works in the "middle."

In a preferred embodiment, the nature and extent of a domain can be determined by several different factors. Since, as shown in FIG. **5**, the domain communications server can connect the intranets of several different companies or institutions in ways that allow them to operate together very closely, it is anticipated that one company might operate the domain communications servers for an industry segment, while the companies that are part of that segment would have their own client side communications servers. In FIG. **5**, for example, domain communications server A1 might be a computer system for the investment banking industry segment of the financial industry, located at applicant's Assignee's corporate headquarters, while clients C1–C8 might be investment banks and investment management firms.

As will be apparent to those skilled in the art, however, the industry segment might be law or automotive manufacturing or pharmaceuticals, or any of a number of other major or minor industry segments. If the industry segment is automotive, for example, domain communications server A1 might be operated by a service company, so that automobile manufacturers might be able to communicate closely with suppliers and dealers. Still using FIG. **5**, in this example, clients C1 and C2 might be competing automobile manufacturers and clients C3–C5 might be major parts suppliers, while clients C6–C8 might be dealers. In a preferred embodiment of the present invention, manufacturer C1 could communicate closely with suppliers C3, C4 and C5, in a secure fashion, while manufacturer C2, its competitor, is also communicating closely with them, as well.

Still in FIG. **5**, in a preferred embodiment of the present invention, it should be noted that domain communications server A1 and client side communications servers C1–C8 can each start up and shut down independently of each other. For example, the computer **20** which is part of domain communications server A1 might be booted (started up) by itself, without any communication with the client side communications server. When that happens, domain communications server A1 initializes itself and checks to see if there are any messages for it. If not, it will wait until one is received. Each of the client side communications servers C1–C8, may have their computers boot at different times, too. For example, if client side communications server C8's computer **20** is booted before domain communications server A1's computer is booted, client side communications server C8 initializes itself, and attempts to register itself with domain communications server A1, by sending messages containing the appropriate Uniform Resource Locator(s) (URL(s)) described below, for registration to domain communications server A1. If domain communications server A1 has not yet been booted, these messages will be queued by client side communications server C8 and sent again later, in a preferred embodiment. As will be apparent to those skilled in the art, the registration messages could simply be regenerated at intervals, instead of queued.

In a preferred embodiment, the URLs sent by the client side communications server and by the domain communications servers usually contain at least the name of a function to be performed by the recipient, such as registration, in this case. In many cases they also point to or include an object.

Also in a preferred embodiment, client side communications servers that are in the same firm or client can communicate with each other, even if the domain communications server A1 normally used by that client is not active. Returning briefly to FIG. **1***a*, to illustrate this, for client C2, client side communications servers C2-HK, C2-NY and C2-LN can all communicate with each other inside client C2's Wide Area Network C2WAN even when domain communications server A1 is not yet operational.

That is, internal users and groups that have been established according to the method and apparatus of the present invention and authorized to create and view content can communicate with each other using the present invention and its organizing and indexing features even when domain communications server A1 is offline or down. Messages that would normally have been sent by the client side communications servers at client C2 to domain communications server A1 are queued and sent whenever domain communications server A1 is started up. As will be apparent to those skilled in the art, other methods of providing for the communication between the client side communications servers and the domain communications server could be used, if desired, such as requiring the domain communications server to be operational before the client side communications servers are allowed to communicate with each other.

"Client side communication server startup

Turning now to FIG. **14**, an overview flow diagram of the initialization of a client side communications server is shown. At step **400**, the user initates the startup of a new client side communications server. At step **402** a server shared object is created, followed by client side server shared objects, admin server shared objects, tools server shared objects and agent server shared objects as created in steps **404**, **406**, **408**, and **410**, respectively. Once the shared objects have been created as shown in Step **412** of FIG. **15**, processing continues. In FIG. **15**, at step **414** the URL's used by the client side communications server (the principal ones of which are listed in FIG. **8***b*) are registered with web server WS. Next, workgroup objects are created at step **416**.

Another view of this is shown in the flow diagram of FIG. **16**. As seen there, at step **420**, a client side communications server is initialized, then, at step **422**, it registers its URLS with the web server WS executing on its computer **20**. At step **424** the client side communications server calls the Register function on the domain communications server. At step **426**, the domain communications server calls the adhocContentProducers functions on the client side communications server, and then, at step **428**, the client side communications server calls the Producers function on the domain server. Next, at step **430**, the domain communications server calls the adhocContentConsumers function on the client side communications server, and finally, the client side communications server calls the indexes function on the domain communications server.

Domain communications server startup

With reference now to FIG. **20***a*, a very simplified overview of the initialization of the domain communications server is shown (more detail is provided below.) At step **500**, the domain communications server initializes itself and the dynamic client registry **06**. Next, the domain communications server checks, at step **502** to see if there are any messages (such as requests to register coming from a client side communications server). If there are no messages, the domain communications server could wait until there is some activity. (As will be seen below, in a preferred embodiment, the domain communications server actually checks periodically to see if client side communications servers are still active.)

Still in FIG. **20***a*, if a message has come in, such as a request from a client side communications server to register itself with the domain communications server, the domain communications server handles the message at step **504**, in the manner indicated by the message itself, as described in more detail below.

Client side communications server dynamic group registry

Referring now to FIG. **20***b*, a flow diagram depicting the process used by a client side communications server to initialize or update the dynamic group registry **07** is shown. Starting at step **800**, a user would give the group to be entered into group registry **07** a name, then, at step **805**, indicate whether the access type for this group is to be common or restricted (these terms will be described in more detail below.) Next, at step **810**, the client side communications server checks to see whether this group will want base content (identified by subject), in which case at step **815** base content is selected, or adhoc content (identified by source) in which case adhoc content will be selected. As described in more detail below, other types of content are also used in a preferred embodiment—mixed content and nondecoupleable mixed content, as well as system content. Processing for these is similar. As will be apparent to those skilled in the art, content can be grouped in any of a number of ways without deviating from the spirit of the present invention.

Still in FIG. **20***b* at step **825**, the client side communications server configures a content list. In one preferred embodiment, the client side communications server next checks to see, at step **830**, whether a rolling link format is desired or a fixed link one, and at steps **875** or **840** update the content list to list a title and indicate the number of rows to display according to the format. Then, at step **845**, the user indicates the subject and/or the source of the content to be displayed on this content list in this group in dynamic group registry **07**. At Step **850**, the user chooses the authoring members, at step **855** the viewing members and finally, at step **860**, the user can review the group's configuration.

FIG. **21** is a flow diagram showing how the client side communications server determines whether to create client side communications server shared objects (compiled C++ code) to communicate with other client side communications servers either inside (see steps **905** and **910**) or outside (see steps **915** and **910**) its own firm.

FIGS. **22** through **26** are block diagrams of the interactive screen displays created by the present invention for use by a browser BR at a terminal. These figures are illustrative of the steps described above to create groups on the dynamic group registry **07** in a preferred embodiment of the present invention. FIG. **22**, for example, shows how a user might enter a new user's name, and FIG. **23** illustrates how account information specific to that user can be entered, such as username, and so on.

FIG. **24** shows how the client side communications server begins the process of establishing authoring rights (described below in more detail.) FIGS. **25***a* and **25***b* illustrate an example of one way of organizing content geographically, by global region. As is indicated, the user is asked to indicate his or her selections for these choices. Next, in FIG. **25***c*, sample distribution options are shown. Next, in FIG. **26***a*, the client side communications server allows the user to specify whether the new group member should have redistribution authorization (this is described below in more detail.) If this is to be allowed, the user can specify, as shown in FIG. **26***b*, whether this redistribution right extends only to internal members on either a limited or unlimited basis or to external members, on either a limited or unlimited basis.

Next, turning to FIG. **26***c*, a sample screen display asking whether the new member is to have access to any restricted groups is shown. If the user said yes to that screen, FIG. **26***d* shows how the user might be asked to specify to which restricted groups the new member should have access, and also in what capacity—that is, as a viewer or a contributor or a redistributor or an administrator, or all or some combination of these.

In FIG. **26***e*, a sample screen display asking the user to specify what kind of external group access the new user should have is shown. Relating back to the example of the financial community, it can be seen in the display shown in FIG. **26***e* that it is a very simple matter to "permission" a new member to create and distribute documents to other participating companies—firms C1, C2 and C3. in the same intelligent extranet.

Once the user has finished answering the questions about adding a new member, the client side communications server completes the updating of the dynamic group registry **07** to reflect these changes.

As will be apparent to those skilled in the art, a client side communications server can be a self-sufficient entity, used simply to provide a much better and simpler intranet management tool inside a corporation than presently exists. The client side communications server also acts as a powerful publications control. By simply answering the questions brought up on the screen displays as new members and groups are added to the client side communications server the user is able to organize, index, and control more than ever before the creation and dissemination of information amongst the individuals and groups known to the client side communications server.

Returning to the financial community example shown in FIG. **1***a*, if this client side communications server is installed at bank C2, (of FIG. **1***a*), which has branches in Hong Kong, New York, and London, it can replicate itself so that there is a client side communications server C2-HK in Hong Kong, a client side communications server C2-NY in New York, and a C2-LN in London. The procedures described above can be used to manage communications amongst all three internal sites in a way that provides organized, fast access to the information created inside bank C2. As will be seen from the discussion of indexing below, a viewer in London can ask for and see indexed information he is authorized to receive, comment on it internally, and have those comments reviewed by his peers in Hong Kong and New York, if he has those kinds of redistribution rights.

Now turning to FIG. **6***a*, typical client information kept by a domain communications server A1 in a dynamic client registry **06** is shown. Dynamic client registry **06** includes a domain clients table **60** (here shown in abbreviated form), a domain clients sources list **66**, domain client objects table **62** and domain client objects destination list **68** and an index table **64**, as well as a channels table **61**, a channel content table **63**, and a channel template table **65**. A domain clients table **60** lists all the client side communications servers within the given domain. As noted above, a client may have more than one client side communications server (for example, where the client has multiple sites across the world, as in the case of Clients C2 and C4 shown in FIG. **1***a*.)

Consequently, as shown in FIG. **6***b* in a preferred embodiment, an entry in a domain clients table **60** of dynamic client registry **06** includes, for each client server, the unique client side communications server name **60***a*, an indicator **60***b* which shows whether the server is active or not, a unique firm identifier **60***c*, which indicates the name of the company or firm or client that owns this server, a firm logo **60***e*, which can be reproduced in visuals associated with this firm and a domain client side identifier **60***f*, which, in a preferred embodiment, is the server location within the firm's intranet, which, when combined with the firm iden-

tifier field 60c, creates a unique key. As will be apparent to those skilled in the art, more or less or other information about each client side communications server could also be included here, if desired. For example, in addition to a logo, perhaps an address of the firm's Web Page on the World Wide Web could be included. Alternatively, a logo field or the hypothetical web page address field, for example, could be eliminated without deviating from the spirit of the present invention.

Returning to FIG. 6a, in a preferred embodiment, each entry in the domain clients table 60 is linked to a domain client sources list 66 (also shown here in abbreviated form.) A domain client sources list 66 is a list of all connections between client side communications servers within the domain. Still in FIG. 6a, and using the automotive domain example, if competitors C1-PA and C2 are two of the competing manufacturers and C3 is a parts supplier to both of them, then domain client sources list 66 might be structured as shown. At column 66a, unique firm identifiers, DSCfirmID's are listed next to each source DSCsourceFirmID in column 66b. Thus, company C1-PA can receive content from itself, C1-PA, as well as from supplier C3. Note that company C1-PA is not shown as being authorized to receive any source information from its competitor, C2. Similarly, company C2 can receive source content from itself and from supplier C3, but not company C1-PA. This illustrates how the virtual pipes described above are formed.

Still in FIG. 6a, a domain client objects table 62 is shown, as well. The domain client objects table contains all distributed objects received by the domain communications server from any valid client side communications server within the given domain. Upon receipt of such an object, the domain communications server notifies all client side communications servers having authorized access to that object that an object is available to be retrieved. Each entry in the domain client objects table 62 points to a different domain client object destinations table 68 that lists all the client side communications server sites that are authorized and destined to receive the domain client object. In a preferred embodiment, a domain client Iohandle is a large object handle within the domain client objects table 60 that points to the distributed content that will be sent to the client side communications servers. Referring briefly to FIG. 6b, with a domain client object destinations table entry, the domain client target table 68a lists all the server names of the client side communications servers to which this object is destined. The domain client received column 68c contains, for each targeted client side communications server, the time the object was received by the domain communications server. The domain client processed table 68d contains, for each client side communications server, the time the object was processed by the client side communications server. In a preferred embodiment, this field is initially set to Null and updated when the client side communication server retrieves the domain client object.

Returning again to FIG. 6a, in a preferred embodiment, an object can be any type of information to be transmitted to a client. For example, in the investment banking domain, it might be the morning analysis prepared by an investment bank C1-PA for its clients. At its simplest, object 62 might be in simple ASCII text format, or in the universally readable Adobe Acrobat™ PDF format, or in a format unique to a word processing program such as Microsoft Word™ or Corel's Wordperfect™, or a spread sheet program such as Microsoft's Excel™ or IBM's Lotus 1-2-3™ spreadsheets. Similarly, if a firm already has existing HTML formatted pages, these, too can be objects. It should be noted that in a preferred embodiment of the present invention, the objects which are the content of all the communications can for all practical purposes, be in any format. For example, an object might also be as complex as a full color movie with sound, or a CAD/CAM VLSI drawing of a chip set or engineering drawings of an automobile under development.

In the latter examples of CAD/CAM or engineering drawings, which are also usually highly confidential, it can be seen that the present invention allows a company such as an automotive manufacturer to enable its engineering department to work very closely and yet securely with subcontractor engineering companies by exchanging and annotating drawings as a project progresses. Since a preferred embodiment of the present invention uses secure socket technology for transmission, objects that are transmitted will be encrypted by the secure socket technology, at the providing client side communications server, and decrypted at the receiving client side communications server of only those clients authorized to receive them. In a preferred embodiment, this encryption and decryption of transmitted objects is an automatic byproduct of the use of secure sockets technology and its equivalents or improvements. As will be apparent to those skilled in the art, other encryption techniques could be used instead of secure socket or similar technologies. For example, techniques such as direct encryption using software such as PGP, which is based on the RSA encryption algorithms could be used.

Still in FIG. 6a, domain client objects table 62 is shown containing an ASCII text format report 62a, a slide presentation 62b, a word processing document 62c, a movie 62d, and multiple reports 62e. In a preferred embodiment of the present invention, objects are represented either as text or as files. Thus the first item, ASCII text format report 62a will be handled by the present invention as ASCII text. All the other objects in domain client objects table 62 are handled as files. In the case of the slide presentation 62b, for example, it is transmitted to domain communications server A1 as a file, and from there it is received by all the appropriate client side communication servers as a file, too. When it is accessed by a terminal T at a client site, the terminal must have the appropriate applications program for viewing that type of file available either at that site or at the server for that site.

So, if slide presentation 62b of FIG. 6a was created using Microsoft's Powerpoint™ software, the viewer at the client site must be able to use Microsoft's Powerpoint software at his or her site to view the slides (usually this can be done by having a copy of the application software installed at the terminal, if it is a computer, or on the server serving this terminal). This is usually not a problem but an advantage, since for many business personal computer users there are commercially available products which have achieved near de facto standards status, such as word processors, spreadsheets and presentation software. As will be apparent to those skilled in the art, the present invention allows the users to continue using these "standard" products and even specially developed software programs that operate on files. This is significant for many users with a major investment in existing application software and prior developments.

Still in FIG. 6a, an indexes table 64 is also shown as part of dynamic client registry 06. The indexes table 64 is used to organize the domain's content. The values of the indexes are the same for a given domain. As shown in FIG. 6a, illustrative indexes for the investment banking market segment domain are shown, including Mortgage, Agency, New Issue, Research and other indexes that might describe the content of the domain in an orderly fashion.

Referring now to FIG. 6*b*, an index entry is shown having an IndexID **64***a*, an IndexDescription **64***b*, an IndexBasetype **64***c*, and an IndexParentID**64***d*. Index ID**64***a* is the unique identifier of the index. In a preferred embodiment, IndexBaseType **64***c* is used to determine which type of format (text or file) is used as the default format for that index, but all content for that index can be of both types. IndexDescription **64***b* is the descriptive label applied to the index. And IndexParentID **64***d* is the IndexID of this entry's parent.

Turning back to FIG. 6*a* again, in indexes table **64** it can be seen that Index **11**, New Issue, is a "child" of index ID **1**, Mortgage. And further, index **61**, Education, is a child of New Issue. In a preferred embodiment, indexes can have as many levels as needed.

Still in FIG. 6*a*, in a preferred embodiment, channels table **61**, channel content table **63** and channel templates **65** are also created. In a preferred embodiment of the present invention a channel is a combination of various indexes. For example, one channel might consist of all the indexes that are likely to contain hot news items. Another channel might be composed of those indexes that are most frequently used. As will be apparent to those skilled in the art, the number and content of such channel groupings might vary from industry to industry or over time. As can be seen, then channels table **61** is used to organize the domain's content into related categories. These channels are then passed to client side communications servers upon startup and are used during the setup and administration of groups. In an alternative preferred embodiment, this function can be included in the client side communications server or a client side communications server using a domain communications server for internal purposes.

With reference now to FIG. 6*b*, channels table **61** contains a channel id field **61***a*, a channel name field **61***b*, and a channel description field **61***c*. For hot news items, a channel id of 1 might be assigned, with a channel name of "Hot News Items" and a channel description such as "hot news from the New Issue, Mortgage and Research indexes." As will be apparent to those skilled in the art, if the domain is in another industry segment, such as automobile manufacture, the indexes and channels will refer to different topics and combinations of topics. In a preferred embodiment, channel content table **63** is used to determine the content that makes up the channel. Again, in FIG. 6*b*, channel content table **63** contains channel id field **63***a*, channel content id **63***b*, which is paired with the channel id field **63***a* to create a unique key, a channel type field **63***c*, which will indicate whether the content is base content or ad hoc content or other types (as described below), and a channel content source field **63***d*, which indicates one or more sources for the content for that channel. In a preferred embodiment, a channel template table **65** is used to organize the channel's content. Each template contains a channel template id field **65***a*, a channel template field **65***b*, which is paired with channel template id field **65***a* to create a unique key, a channel template attributes field **65***c* which defines the attributes of the content list, and a channel template sources field **65***d*, which refers to id's found in the channel content table **63**.

With reference now to FIG. 7*a*, illustrative structure and contents of dynamic group registry **07** of the present invention are shown. In a preferred embodiment, dynamic group registry **07** is used at each client to handle all the content that is produced internally at that client as well as the content that is produced throughout the domain that is received by that client. As noted above, each client side communications server must use the domain communications server in order to communicate with another firms's client side communications server.

As shown in FIG. 7*a*, the principal entry in client side dynamic group registry **07** is group table **70**. Group table **70** is a list of all groups that have been established within a particular firm or client. Note that, while the term workgroup may occur in the figure or this text, the term group is generally preferred and can be read interchangeably. A group is used to organize the type and source of the content (as specified in content table **90** and ad hoc content table **94**) available to that group and creates the respective content lists for the group as information is received. CSContent table **90** organizes content by the object id **90***a*, the author id **90***b*, the content link object id **90***c*, the content headline **90***d*, the content filename **90***f*, the content information **90***g*, the content timestamp **90***h*, the content origin site **90***i*, the content origin object id **90***j*, and the content data attribute **90***k*. This is automatically added as it is produced to all groups that have requested that specific type of content. Ad hoc content table **94** organizes ad hoc content by the source of the information and is added to a group when the author (or publisher) directs the ad hoc content to that group.

As can be seen in group table **70** of FIG. 7*a*, in a preferred embodiment there are two types of groups—common and restricted. A group that is common is accessible by all viewers within the client or firm and can only take base content. Restricted groups are viewable only by members of the group as specified in the workgroup view members table **82**. A restricted group allows for both base content and ad hoc content as well as other types of content that may be developed. Ad hoc content can only be added to the group by the group's author and publisher members (workgroup author members table **80** and workgroup view members table **82**.)

Still in FIG. 7*a*, in group table **70**, it can be seen that in a preferred embodiment, all groups have a community, indicated in the field WGcommunity, in which they are "known." The community is used to 'advertise' to other groups within the firm or client. The community contains producers of content that is advertised either internally (to the firm or client only) or externally to the entire domain. An internal group is not known outside of the firm or client. An external workgroup is known inside, as well as outside, the firm or client. If a workgroup is set as a producer, by using a value of 2, in a preferred embodiment in the WGcommunity field of group table **70**, the group is advertised as a source of ad hoc content for other groups.

Still in FIG. 7*a*, in a preferred embodiment, groups that are solely internal to the firm or client are so indicated by using a 0 in the WGcommunity field of group table **70**. As shown in group table **70**, workgroup WG1, a firm wide group, is the only internal group listed in this particular example. External groups are indicated by the use of a 1 value in the WGcommunity field of group table **70**. In setting the values for the WGcommunity field, a preferred embodiment exclusive OR's the values to determine all choices. In a preferred embodiment, the WGgroupID's are always unique since they are composed of a firmid, a site id and a number. As will be apparent to those skilled in the art, any of a number of ways can be used to create unique group or workgroup ids.

Still in FIG. 7*a*, workgroup base content table **74** is used to determine the base content the respective workgroup is interested in receiving when such content is produced, either internally or externally. Base content is selected by the content type, (the index value WGBCindexID) and can be received by both common and restricted workgroups. Upon receiving a particular content type, the workgroup's content list headlines are updated.

Workgroup master lists table **76** provides descriptive text information about the list.

And again in FIG. *7a*, workgroup ad hoc content table **78** is used to determine the ad hoc content the respective workgroup is interested in receiving when such content is produced, either internally or externally. Ad hoc content is determined by the source of the content as opposed to the index of the content. The sources for ad hoc content are those groups that have "advertised" themselves, as described above. Mixed content is a combination of both adhoc and base content.

Also in FIG. *7a*, workgroup author members table **80** contains the list of authors and the groups that have specified they will allow this author to add ad hoc content to the respective group. The authors are chosen from the list of authors within a firm or client. The groups are also from the list of groups within the firm or client. Entries into this table will only exist for groups that have specified that they are producers of new content. In the example of the workgroup author members table **80** shown here, there is a workgroup author identifier WGAMauthorID. This field is linked to the ViewernD of the viewers table described below. A workgroup author members group identifier WGAMgroupID identifies the unique group within this client to which this member belongs. This value is linked to the WGgroupID of group table **70**.

In the example shown in FIG. *7a*, the same author, tburns, is listed as a member of workgroup **2** WG2 and workgroup **3** WG3. The field workgroup author member restrict WGAMrestrict is, in a preferred embodiment, a boolean value that is set to true if the author must have been individually "permissioned" to send to each of the workgroups that consume this workgroup's content. If this value is set to false, the author can send to any workgroup that consumes this group's content.

Still in FIG. *7a*, workgroup view members table **82** contains the list of viewers for the workgroups. Only viewer members of a workgroup have view access to a workgroup's content. All viewer members of a workgroup are selected from the list of valid viewers for the firm or client. A workgroup administrator (as specified in the WGVMadmin field) is allowed to determine members and the attributes of any members for the group. Distribution of the group's content is controlled by the viewer attribute field WGVMattribute. In the example shown here, viewers tmalone and tburns of workgroup **2** WG2, are administrators for workgroup **2** WG2. In a preferred embodiment, there are five values that may be set for the viewer attribute field WGVMattribute. These are:

Distribute None 0

Distribute Internal 1

Distribute External 2

Distribute Internal Restricted 4

Distribute External Restricted 8

Also in a preferred embodiment, these values are exclusive OR'd to determine all choices. As will be apparent to those skilled in the art, various other ways could be used to indicate the attributes of a given member.

In FIG. *7a* again, address book table **84** contains a list of group(s) to which a given author or publisher is permissioned to distribute content as a member of the address book group identifier ABgroupID. That is, ABgroupID is the unique ID of the workgroup that has permissioned the indicated author or publisher to distribute the workgroup's content. This value is linked to the GroupID of the group table **70**. Address book destination field ABdestination contains the firmID and the workgroup ID of the group to which the author or publisher can distribute content. Address book attribute field ABattribute of address book table **84** is, in a preferred embodiment, a Boolean value set to true if the workgroup allows this author or publisher to add a note to the content, and to false if not.

Remaining in FIG. *7a*, author rights table **88** is used to determine the type of content a given author can create. The table contains three fields: author identifier AuthorID, author index identifier AuthorindexID, and author attribute AuthorAttribute. In a preferred embodiment, the content type (as indicated by the author attribute field) that an author can create is determined by the firm or client. This allows the client to define and implement its own internal policies for such matters, rather than obligating the firm to abide by policies established by others or the constraints of a particular external system. All authors must be valid viewers of the firm or client.

In a preferred embodiment, all authors have a minimum author attribute of internal broadcast for the specified content type. In a preferred embodiment, the possible values are:

Internal Broadcast 0

Internal Selective 1

External Broadcast 2

External Selective 4

These values are exclusive OR'd to determine all choices, in a preferred embodiment. As will be apparent to those skilled in the art, other ways of defining and implementing such values could be used.

Still in FIG. *7a*, viewers table **86** contains the list of all valid users or viewers at a given client or firm. This table is used to determine if the specified user or viewer is a member of the client's intranet and access to that client's intranet will only be allowed if the user or viewer is listed in this table. Users or viewers entered in this table are allowed access to all client or firm workgroups that are common, as specified in the WGmembership field of group table **70**.

Turning now to FIG. *8a*, and back to a discussion of domain communications servers, a list of the principal domain communications server URLs is shown. As mentioned above, a domain communications server is implemented, in a preferred embodiment, using AOLserver software, because of its ability to dynamically load shared objects when spawning a virtual server. In a preferred embodiment of the present invention, object-oriented programming techniques are used, since they allow the creation of procedures for objects whose exact type is not known until actual running of the program. Object oriented techniques also permit the system implementer to define and use shared objects—compiled C++ code that is called by more than one function or program. In the AOLserver, for example, which shared object to load is specified to it within an initialization file used when starting AOLServer's NSD process. In a preferred embodiment of the present invention, the shared object named 'domainserver.so' (also known as 'domainserver.dll' for NT versions), is designated as the shared object to be loaded when the AOLserver is started up.

In a preferred embodiment, the AOLServer Applications Programming Interface (API) is used because it is an interface which allows for the development of shared objects which can be loaded by AOLServer. The API is ANSI-C, a standard programming language interface and therefore requires C++ wrapper functions in order to bridge between the C based AOLServer and C++ based domain communications server's objects. There are two wrapper functions defined by the domain communications server, named "go"

and "stop". As will be apparent to those skilled in the art, any server or program which provides similar functionality could be used instead.

The AOLserver NSD parent process, when starting any virtual servers, will look for a function named Ns-ModuleInit which (per AOLServer documentation) must be defined for all shared objects that are loaded and contain any required start-up code. The domain communications server of the present invention's Ns ModuleInit function has two responsibilities, first to register the shutdown procedure "stop" that will be called by the parent NSD when shutting down secondly, to call the C++ wrapper function "go" which creates an instance of a DomainServer object. The shutdown procedure "stop" is responsible for cleaning up the Domain-Server object initiated by calling "go".

As described above, a domain communications server according to the method and apparatus of the present invention is responsible for distributing and collecting content from various client side communications servers within its domain. It will also control which client side communications servers may communicate with each other and act as "switching point".

The creation of a domain communications server is all that is required by the wrapper function goo. When creating a domain communications server object the following steps ate taken:

First the domain communications server saves local variables that specify (among other things) the "recognized" name of the virtual server and access to the underlying database manager.

Next, the available categories used by the domain to organize all of a domain's content are loaded from the table Indexes, (see FIG. 6*b*) and are stored internally as Index Objects.

Third, the domain communications server determines all valid domain clients that are known to be within this domain, their network addresses, and which clients can communicate with each other—the virtual pipes (as shown in FIG. 1*a*.) Valid clients, client side communications servers and their respective virtual pipes are found within the tables DomainClients and DCClinetSources and are stored internally as DomainClient Objects.

Fourth, the domain communications server will determine the predefined templates used by the domain's clients when creating new groups at the client side communications server sites. These templates are used to organize the domain's Index Objects and network producers into a more organized format when setting up what is consumed by the client side communications servers. The information for these templates are found within the tables Channels, ChannelContent and ChannelTemplates shown in FIG. 6*a* and are stored internally as Channel Objects.

Lastly, the domain communications server will register with the AOLserver parent NSD process the URLs (Uniform Resource Locators) shown in FIG. 8, that the domain communications server will handle and the corresponding callback function in the domain communications server that the AOLserver parent NSD process should call when any such URLs arc requested. These registered URLs are used to communicate between the domain communications server and any valid client side communication server within the domain. Information is passed between client side communications servers and the domain communications server via these registered URLS as a result. All information passed is in a form known in the art as "persistent objects" (i.e. objects that can restore themselves from stream form) and are retrieved and/or distributed via a respective URL.

The following is a list of the URLs and allowable methods that are registered and the event class associated with each in a preferred embodiment:

URL	Method	Event Class
/Register	GET	CSS Registration
/Indexes	GET	CSS Registration and Registry Change
/Consumer	GET	CSS Registration and Registry Change
/Producers	GET	CSS Registration and Registry Change
/Channels	GET	CSS Registration and Registry Change
/Unregister	GET	CSS Deregistration
/Status	GET	Status
/DistributeObject	GET	Content Replication
/CollectObject	GET	Content Replication
/Refresh	GET	Registry Change
/FirmLogo	GET	CSS Registration
/InternalServers	GET	CSS Registration
Status/ReloadIndexes	POST	Status
/Status/CSSStatus	GET	Status

After startup, the domain communications server will then schedule a timer procedure to check every 60 seconds the current status of each valid client side communications server in its domain and "ping" those for which no activity has been recorded within the last five minutes. In a preferred embodiment, a ping is simply a message requesting a status be returned from the client side communications server.

An example of a domain communications server attempting to ping a client side communications server might look like:

https://validCSS.com:84/Ping

In response to this URL, the client side communications server in a preferred embodiment will return the HTTP status code **200** signaling the domain communications server. The pinging of a client side communications server will determine its current status and ensure that it is up, running, and registered with the domain communications server.

If a client side communications server responds to a ping message the domain communications server will check to see if the client side communications server is already currently registered and if it is not, it will ask the client side communications server to attempt to register at this time (or re-register if the client side communications server has been running longer than the domain server). If the client side communications server is currently registered the domain server will note the last time it contacted this client side communications server and will not ping it again unless a lag of 5 minutes or more occurs before contact is re-established. If the client side communications server does not respond to the ping, the domain communications server will note the client side communications server as not registered and will attempt to ping this client side communications server every 60 seconds until a response is finally returned.

Once initialized, the domain communications server simply responds to events that occur within the domain The domain communications server is responsible for verifying events as valid as well as notifying any client side communications server of the effect of such event. This is done through the handling of the URLs specified above.

The following is a list of the event classes and the corresponding actions that can occur within the domain:

1. Client side communications server registration.

The registration of a client side communications server with the domain communications server is started when the client side communications server requests the "Register" URL from its domain communications server (the address of the domain communication server is found within the initialization file used when starting the domain communications server). In a preferred embodiment, the format of this URL is "DomainServer:/Register?ID=CSS" where Domain Server is the protocol, machine name (and port) to request from and ID variable is the CSS's protocol, machine name (and port). Since the protocol is specified in this manner, it is trivial for the domain to use either standard HTTP or the SSL layer. In an alternative preferred embodiment, the present invention implements the X.500 Lightweight Directory Access Protocol (LDAP.)

An example of a client side communications server attempting to register might look like:

https://myDomainServe.com:81/Register?ID=https://validCSS-.com:84

The domain communications server will then verify that the location specified by the ID is a valid location for this domain by checking its internal collection of DClient Objects (loaded at startup). If the request were validated, the domain communications server would return to the client side communications server the "virtual pipes" available to this client side communications server by passing a list of ClientSource Objects, from dynamic client registry **06**. It can be seen that dynamic client registry **06** serves as the repository that enables the virtual community of intranets to be formed.

In a preferred embodiment, validation requires that the domain communications server request the ContentProducer and ContentConsumer Objects from the registering client side communications server's site. The response from the client side communications server is the stream form (streaming is an input/output format for data used in most open systems) of the respective objects. The objects are recreated at the domain communications server and collected locally. It is the domain communications server's responsibility to determine which ContentProducers and ContentConsumers are available to which client side communications server by checking with the list of "virtual pipes" already established.

An example of the domain communications server requesting a registering client side communications server for its ContentProducer objects is:

https://validCSS.com:84/AdhocContentProducers

An example of the domain communications server requesting the client side communications server for its ContentConsumer objects is:

https://validCSS.com:84/AdhocContentConsumers

In a preferred embodiment, the registering client side communications server will then request additional information about the domain using the other URLs handled by the domain communications server, as shown in FIG. **8***a*. All potential recipients are validated prior to returning any requested information. This information will include the following:

* The list of Index Objects used by the domain is requested by the Indexes URL **106** of FIG. **8***a*. Index Objects are used to organize content by subject matter (also known as base content) within a given domain and are hosted centrally by the domain communications server in dynamic client registry **06**. Index Objects are used by the client side communications server as part of its dynamic group registry **07**.

The response to the Indexes URL **106** of FIG. **8***a* causes the domain communications server to iterate through its collection of Index Objects and return them in stream form. The stream form is translated back into Index Objects at the client side communications server site and collected locally.

An example of a client side communications server requesting the indexes might look like:

https://myDomainServercom:81/Indexes?ID=https://validCSS-.com:84

* The list of InternalServer Objects is collected by requesting the InternalServers URL **126** of FIG. **8***a*. InternalServer Objects are used by a client side communications server in determining the sites of all other client side communications servers considered to be of the same firm and hence the client side communication server's responsibility to replicate to and from.

The response to the InternalServers URL **126** causes the domain communications server to ask the validated DomainClient to iterate through its collection of VirtualServerObjects and return them in the stream form. The stream form is translated back into InternalServer Objects at the client side communications server site and collected locally. Only those client side communications servers of same firms are returned (this includes the requesting client side communications server as well).

An example of a CSS requesting the internal servers might look like:

https://myDomainServer.com:81/InternalServers=https://validCSS-.com:84

* The list of ContentProducer Objects is retrieved from the Producers URL **110** of FIG. **8***a*. ContentProducers are used by the client side communications server as part of the dynamic group registry **07** as available sources of adhoc and mixed content. ContentProducer objects contain the information about the producer and what it Produces. It is the domain communications server's responsibility to determine which ContentProducers are available to the requesting client side communications server by checking with the list of virtual pipes already established.

The response to Producers URL **110** of FIG. **8***a* causes the domain communications server to return in stream form the ContentProducers available to the requesting client side communications server from all other client side communications servers within the domain. For sites of multiple internal servers the stream includes internal producers as well. The stream form is translated back into ContentProducer Objects at the client side communications server site and collected locally.

An example of a client side communications server requesting the ContentProducers might look like:

https://myDomainServer.com:81/Producers?ID=https://validCSS-.com:84

* The list of ContentConsumer Objects is requested via the Consumers URL **108** of FIG. **8***a*. ContentConsumers are

used by the client side communications server as part of the "domain registry" as to determine which of its producing groups have consumers requesting their content.

The response to the Consumers URL **108** of FIG. **8***a* causes the domain communications server to return in stream form the ContentConsumers available to the requesting client side communications servers from all other client side communication servers within the domain. For sites of multiple internal servers the stream includes internal consuming groups as well. The stream form is translated back into ContentConsumer Objects at the client side communications server site and collected locally.

An example of a client side communications server requesting the ContentConsumers might look like:

https://myDomainServer.com:81/Consumers?ID=http://validCSS-.com:84

* The list of ClientInfoChannel Objects are collected via the Channels URL **128** of FIG. **8***a*. ClientinfoChannel objects are used as templates for creating new groups at the client side communications server. The InfoChannel object organizes portions of the domain's content into preconfigured views that ease the creation of a group at a client side communications server site.

The response to the Channels URL **128** of FIG. **8***a* causes the domain communications server to return in stream form the Infochannel objects available to the requesting client side communication server. The stream form is translated back into ClientInfoChannel Objects at the client side communications server site and collected locally.

An example of a client side communications server requesting the Channels might look like:

https://myDomainServer.com:81/Channels?ID=https://validCSS-.com:84

* The list of Logo Objects are collected via the FirmLogo URL **116** of FIG. **8***a*. Logo Objects are used to further "brand" content with the originator. The registering client side communications server will request for any Logo Objects it does not currently have a "current" Logo Object". Status of the LogoObject is determined from the list of ClientSource Objects received from the Register URL **102** of FIG. **8***a*.

The response to the FirmLogo URL **116** of FIG. **8***a* causes the domain communications server to return in file form the corresponding LogoObject of the client side communications server specified. The object is stored on the local file system and served whenever content that originated from this source is viewed.

An example of a client side communications server requesting the LogoObjects might look like:

https://myDomainServer.com.81/FirmLogo?ID=https://validCSS.com:84&SRC=https//myCSS2.com.85

The registration of a client side communications server will trigger the domain communications server to notify all of the other client side communications servers (having virtual pipes to the registering client side communications servers) that changes in the dynamic client registry **06** have occurred that may be of interest to them. This will cause all Registry Change events to occur at each respective client side communications server. The changes include the availability of both additional consumers and producers found within the registering client side communications server. The notification is done by requesting the /Refresh URL at each client side communications server and specifying that both the /Producers and /Consumers have changed and should be refreshed at the client side communications servers' convenience (see Registry Changes below for a detailed description). In a preferred embodiment, registration is always initiated by the client side communications server. However, also in a preferred embodiment, the domain communications server may ask any client side communications server at anytime to re-register when the domain communications server deems it appropriate.

2. Client side communications server deregistration

The deregistration of a client side communications server occurs when the client side communications server needs to notify the domain communications server that it will no longer be available to the domain. This event is controlled by the client side communications server and can occur at anytime, but, in a preferred embodiment, will always happen when the client side communications server attempts to shutdown normally. The client side communications server notifies the domain communications server of its desire to unregister via the Unregister URL **104** of FIG. **8***a*. The format of this URL is Domainserver:/Unregister?ID=CSS" where Domain Server is the protocol, machine name (and port) to request from and ID Variable is the client side communications server's protocol, machine name (and port).

An example of a CSS attempting to deregister might look like:

https://myDomainServer.com:81/Deregister?ID=https://validCSS-.com:84

The domain communications server will then verify that the location specified by the ID is a valid location for this domain by checking its internal collection of DClient Objects (loaded at startup). If the request was validated, the domain communications server will return the HTTP status code **200** signaling the client side communications server that it is now considered unregistered by the domain. All content destined for this client side communications server will be queued at the domain communications server until the client side communications server becomes available for the domain again.

In a preferred embodiment, deregistration of a client side communications server may also be initiated by the domain communications server. This can occur when a client side communications server does not respond to the /Ping URL-requested by the domain communications server when a specified amount of time (usually 5 minutes) has expired since last contact from the client side communications server. As will be apparent to those skilled in the art, shorter or longer time periods could be specified for this interval. And similarly, while a preferred embodiment uses this approach, and queues messages for the client side communications server deregistered for this reason, it will be apparent to those skilled in the art that other techniques might be used to cause the messages to be held for the non-responding client side communications server.

In a preferred embodiment, the deregistration of a client side communications server will trigger the domain communications server to notify all other client side communications servers having virtual pipes to the deregistering client side communications server that changes in the domain's registry have occurred that may be of interest to them. The changes include the unavailability of both addi-

tional consumers and producers found at the deregistering client side communications server. The notification is done by requesting the /Refresh URL at each client side communications site and specifying both the /Producers and /Consumers have changed and should be refreshed at the client side communications server's convenience.

3. Content replication

This event occurs whenever content must flow from one client side communications server to another client side communications sever within the domain. It should be noted that in a preferred embodiment, the domain communications server is only responsible for content replication between client side communications servers of different firms (external replication). Content replication between client side communications servers of the same firm (internal replication) is handled directly between the two (or more) client side communications servers themselves.

In a preferred embodiment, there are three types of content, Base Content, Adhoc Content, and Mixed Content, that can be replicated throughout the domain. (Also in a preferred embodiment, there is a fourth type of content known as SystemContent, but this type is not replicated). The type of content is determined by how the content is organized. Base content is content which is organized by topic or subject matter. The subject matter must be defined within one or more of the domain's Index Objects. The origin of the content (i.e. where is was created) is not relevant. Base content may have more than one index associated with it. Adhoc content, on the other hand, is content which is organized by its origin and not its subject matter. The origin must be one or more of the "producing" groups at a given client side communications server within the domain. Mixed Content is a combination of both adhoc and base content. Mixed has both an origin and subject matter associated with it.

In a preferred embodiment, mixed content can be of two separate types, uncoupleable and decoupleable. Uncoupleable mixed content is content that cannot be broken into its base and adhoc parts and must be treated as a whole, whereas decoupleable can be broken into its subcomponents and treated separately. The type of content is determined by the producing client side communications server at the time of content creation.

The type of replication of content can be one of two possible modes, Broadcast or Selective Distribution, and is controlled by the client side communications server based on possible destinations permissioned to the producing individual at the client side communications server site. Broadcasting content has the effect of distributing the (base) content to all other client side communications servers within the domain in which that originating client side communications server has established virtual pipes. Selective Distribution will only replicate (adhoc and/or mixed content) to the specified client side communications servers and not the entire domain. In a preferred embodiment, the type of replication specified dictates the type of content being replicated. That is, only Base Content can be broadcasted and only Adhoc and Mixed Content can be selectively distributed.

Flow of content is controlled by the DistributeObject URL **112** of FIG. **8***a* and the CollectObject URL **114** of FIG. **8***a* which are used to collect and distribute content throughout the domain. The type of content is not relevant to the replication process since only client side communications servers need to understand about its type (in order to process it) and the content is encapsulated within a DistributedObject, The DistributedObject consists of four parts, the origin, the target, the data, and the signature. The origin and targets are used by the domain communications server to determine who can receive the object, while the signature and data portions are used by the client side communications server to restore the object in order to process.

In a preferred embodiment, replication is initiated by a client side communications server which has predetermined to distribute content externally. In a preferred embodiment, the client side communications server stores the DistributedObject locally and notifies the domain communications server that a DistributedObject is waiting for delivery at the client side communications server site. The client side communications server specifies a unique identifier to the DistributedObject that the domain communications server should use when collecting the object. The client side communications server does this through the DistributeObject URL **112** of FIG. **8***a*.

An example of a client side communications server attempting to notify the domain communications server to Distribute an Object may look like this:

```
https://myDomainServer.com:81/DistributeObject?ID=https://
ValidCSS.com:84&OID=3043.201e
```

In response to the DistributeObject Url **112** of FIG. **8***a*, the domain communications server will return the HTTP status code **200** signaling the client side communications server that its notification was noted. After the notification is sent, the domain communications server requests the DistributedObject from the original client side communications server using the unique identity previously specified. This is accomplished by requesting the Collectobject URL **114** of FIG. **8***a*.

An example of a domain communications server collecting a Distributed Object from a client side communications server might look like

```
https://validCSS.com:84/CollectObject?OID=3043.201e
```

The response from the client side communications server is the requested DistributedObject in stream form. The stream is captured by the domain communications server and recreated into a DistributedObject at the domain communications server site. The client side communications server will note the time locally that the DistributedObject has been retrieved by the domain communications server for auditing purposes. After receiving theDistributedobject the domain communications server will determine the source and list of possible targets specified. The domain communications server will map the targets with those destinations with which the originating client side communications server has established a virtual pipe. The DistributedObject is then stored by the domain communications server locally in the DCObjects table **62** as shown in FIG. **6***a*, and each validated destination is stored within a DCobjectdestinations table . The domain communications server will then notify each client side communications server destination that a DistributedObject is waiting for it. This is done through the use of the ReceiveObject URL **212** of FIG. **8***b*, a client side server URL. A unique identifier for the specified DistributedObject is passed in the URL and is used by the receiving client side communications server when requesting the object.

An example of a domain communications server notifying a client side communications server that a Distributed Object is available might look like:

https://validCSS.com:84/ReceiveObject?LOH=1010982029384

In response to this URL the client side communications server will return the HTTP status code **200** signaling the domain communications server that its notification was noted. After the notification is sent, the client side communications server requests the DistributedObject from the domain communications server using the unique identity specified. This is accomplished by requesting the CollectObject URL **114** of FIG. **8***a*.

An example of a client side communications server attempting to retrieve a DistributedObject from the domain communications server might look like:

http://myDomainServer.com:81/ CollectObject?ID=https://validCSS.com:84&LOH=1010982029384

The response from the domain communications server is the requested DistributedObject in stream form. The stream is captured by the client side communications server and recreated into a DistributedObject at the client side communications server site. The domain communications server will note the time locally that the DistributedObject has been retrieved by the client side communications server for auditing purposes. After receiving the DistributedObject the client side communications server will determine if further processing is needed and do so accordingly.

4. Dynamic client registry changes

In a preferred embodiment, dynamic client registry **06**, as shown in FIG. **5**, is used to determine what resources are available within a given domain at a given point in time. Dynamic client registry **06** consists of the following four objects: content producers, content consumers, index objects, and channels. Dynamic client registry **06** is dynamic in nature, and changes to it are controlled by the domain communications server and can occur at any point in time. The domain communications server is responsible for notifying all client side communications servers of any changes. This is done through Refresh URL **118** of FIG. **8***a*.

In a preferred embodiment, changes to dynamic client registry **06** occur as a result of four different events. The first is the registering of a new client side communications server within the domain. This will cause the domain communications server to notify all other "interested" client side communications servers of this new client side communications server. Notification consists of the listing of ContentProducers and ContentConsumers that the registering client side communications server contains that are now available to all other existing client side communications servers. It is the domain communications server's responsibility to determine which client side communications servers see which producers and consumers (based on the established virtual pipes). In a preferred embodiment, a client side communications server can selectively choose to notify only a subset of those the domain communications server says are available.

The second event causing a change in dynamic client registry **06** is the opposite of the first, namely the Deregistration of a client side communications server. In either event the domain communications server will notify all relevant client side communications servers of changes to both ContentProducers and ContentConsumers objects.

An example of a domain communications server notifying a client side communications server of a change to dynamic client registry **06**, (specifically ContentProducers) might look like:

https://ValidCSS.com:84/Refresh?URL=/Producers

An example of a domain communications server notifying a client side communications server of a change to dynamic client registry **06** (specifically ContentConsumers) might look like:

https://validCSS.com:84/Refresh?URL=/Consumers

The third type of change to dynamic client registry **06** is a change in the IndexObjects used by the domain. This may only occur through the Status events (listed below) handled by the domain communications server. All valid client side communications servers are notified of this type of change to dynamic client registry **06**.

An example of a domain communications server notifying a client side communications server of changes to dynamic client registry **06**, (specifically IndexObjects) might look like:

https://validCSS.com:84/Refresh?URL=Indexes

The last type of change is a change in the Channels used by the domain. This may only occur through the Status events (explained below) handled by the domain communications server. All registered client side communications servers are notified of this type of change.

An example of a domain communications server notifying a client side communications server of changes to dynamic client registry **06** (specifically Channels) might look like:

https://validCSS.com:84/Refresh?URL=/Channel

In response to this URL the client side communications server will return the HTTP status code **200** signaling the domain communications server that its notification was received by the client side communications server. After the notification is sent, the client side communications server will request the URL (with the required parameters) specified from the domain communications server.

5. Domain Shutdown

This event occurs, when the domain communications server must notify all valid client side communications servers that it is no longer available. This notification is handled via DomainSbutdown URL **130** of FIG. **8***a*.

An example of a domain communications server notifying a client side communications server of the domain shutdown might look like:

https://validCSS.com:84/DomainShutdown

In a preferred embodiment, notification is sent to the client side communications server to allow the client side communications server to begin queuing any content that must be replicated externally until the domain communications server notifies the client side communications server that it has once again become available. The client side communications server may use an alternative domain communications server (if one is specified when starting the client side communications server) until the original domain communications server returns to avoid having to queue any content. Upon return of the domain communications server to the domain, the client side communications server will notify the domain communications server of any content that is currently queued by initiating content replication (event 3 above). Notification of the domain communications server's return is done by the domain communications server requesting each client side communications server to re-register (event 1).

6. Status

In a preferred embodiment, events are handled by the domain communications sever in response to queries about

the current Status of the domain and changes to those portions of dynamic client registry **06** which are centrally hosted by the domain communications server (namely Channels and Index Objects). These events are not available to any client side communications server but only to the domain communications server administrators. Status events will display the current status of all client side communications servers within the domain. The status of a client side communications server includes the currently established virtual pipes to other client side communications servers, the content producers (both internal and external) of a client side communications server, and the content consumers of a client side communications server (as well as what the consumers are consuming). The domain communications server may be told to reload the domain's indexes or channels through these events.

As mentioned, in a preferred embodiment, the system also uses a DistributedObject Object. The DistributedObject is used to replicate data from one client side communications server to another within a given domain. The DistributedObject has four components that make it up: the origin, the target, the signature, and the data (the content) being replicated. The origin and target fields are set by the client side communications server creating the data and evaluated by the domain communications server when determining the possible destinations to which the data may replicate. The signature is used by the receiving client side communications server to reconstruct from the data its original type and to understand how to process it. A DistributedObject is persistent, The origins and destinations are in the form of IdObjects. Multiple destinations and origins may be specified.

In a preferred embodiment, the IdObject is used to determine the destination or source of the domain's content. It is made up of the firm, workgroup, and user identifications and is used by both the domain communications server and client side communications servers to determine the origin or targets for contents This object allows for wild card abbreviations to denote all possible matches and can determine if a given IdObjects is "equal to" another IdObject. Valid Idobjects are in the following form and sample wild card abbreviations for them are shown as:

IdObject	Equivalence
***	all users in all groups within all firms
FIRM.*.*	all users in all groups at a specific firm.
FIRM.GROUP.*	all users in a specific group of a specific firm
FIRM.GROUP.USER	specific user in a specific group of a specific firm

As will be apparent to those skilled in the art, when multiple domain communications servers are used a domain field will be included in front of the firm field.

In a preferred embodiment of the present invention, readership analysis and similar statistical information can be kept, if desired, about the use, copying, accessing or types of information referred to by users. As will be apparent to those skilled in the art, analysis programs to track other types of use, traffic flow, response times, and so on could also be implemented without deviating from the spirit of the present invention.

In a preferred embodiment of the present invention, the system can also provide a distribution summary that, for example, allows a user to review at the end of a day, all the information sent out by the user at the end of a that day, or another time period.

Also in a preferred embodiment, the system provides search features to allow a user to search for information based on the information's type or source or both, as well as such other factors as creation date, publication within a time frame and so on. As will be apparent to those skilled in the art, a number of differing search functions can be implemented without deviating from the spirit of the present invention.

In a preferred embodiment, a producer object (which lists the firm name, the group name, etc.,) also lists the producer individuals with whom a receiving consumer can communicate on a one to on basis. Similarly, the consumer object, amongst other items, includes a list indicating producing individuals to whom it is willing to talk on a one to one basis. This one-to-one communications facility of the present invention permits highly confidential communications to occur on a one-to-one "for your eyes only" basis, if desired. As will be apparent to those skilled in the art, various combinations and variations of this one-to-one relationship management are possible.

Similarly, in a preferred embodiment a comment can be tagged with more than one index value.

While a preferred embodiment of the present invention is implemented as a program written in the C++ programming language and operates on personal computers or workstations using the NT or Unix operating systems, as will be apparent to those skilled in the art, other programming languages and operating systems could be used. Additionally, although the preferred embodiment uses a software program implementation, it will be apparent that some or all of the logic of the present invention could also be embodied in firmware or hardware circuitry.

Those skilled in the art will appreciate that the embodiments described above are illustrative only and that other systems in the spirit of the teachings herein fall within the scope of the invention.

What is claimed is:

1. A publication control system for networks inside the same client entity comprising:

a plurality of publication computers in communications relationship with each other inside the client entity, each of said publication computers having electronic storage media for storing a dynamic group registry thereon and for storing resource locators containing function names thereon, each publication computer further comprising a web server program which, when executed by the publication computer, causes the publication computer to respond to resource locators by calling the function name indicated therein into the publication computer, each publication computer further comprising a database management program for organizing the dynamic group registry;

a client side communications server program stored in each publication computer, which, when loaded by the web server program responding to the appropriate resource locator therefor, is executed by each publication computer, and is further responsive to resource locators directed to the client side communications server program and which directs the database management program in organizing the dynamic group registry;

a domain computer having electronic storage media coupled thereto for storing a dynamic client registry thereon and for storing resource locators containing function names thereon, the domain computer further comprising a web server program which, when executed by the domain computer, causes the domain computer to respond to the resource locators by calling the function name indicated therein into the domain

computer, the domain computer further comprising a database management program for organizing the dynamic client registry;

a domain communications server program which, when loaded by the web server program responding to the appropriate resource locator therefore, is executed by the domain computer, and is further responsive to resource locators directed to the domain communications server program and directs the database management program in organizing the dynamic client registry;

a domain communications resource locator list stored in the domain computer and each publication computer that causes predetermined functions to be selected for execution in the domain communications server in the domain computer;

a client side communications resource locator list stored in the domain computer and each publication computer that causes predetermined functions to be selected for execution in the client side communications server in each publication computer so that communications between the domain computer and each publication computer cause the selected predetermined functions to be executed dynamically in order to manage information communications between the domain computer and each publication computer.

2. The apparatus of claim 1, wherein the domain communications resource locator list includes a register domain communications resource locator for causing the domain communications server program to direct the database management program to register a client side communications server program in the dynamic client registry.

3. The apparatus of claim 1 wherein the selected predetermined functions further comprise one to one communications functions.

4. The apparatus of claim 1 wherein the selected predetermined functions further comprise group to group communications functions.

5. The apparatus of claim 1 wherein the selected predetermined functions further comprise site to site communications functions.

6. The apparatus of claim 1 wherein the selected predetermined functions further comprise publication distribution functions.

7. The apparatus of claim 1 wherein the selected predetermined functions further comprise publication editing functions.

8. The apparatus of claim 1 wherein the selected predetermined functions further comprise publication classification functions.

9. A computer implemented publication control method for networks inside the same client entity comprising the steps of:

establishing a plurality of publication computers in communications relationship with each other inside the client entity, each of said publication computers having electronic storage media for storing a dynamic group registry thereon and for storing resource locators containing function names thereon, each publication computer further comprising a web server program which, when executed by the publication computer, causes the publication computer to respond to resource locators by calling the function name indicated therein into the publication computer, each publication computer further comprising a database management program for organizing the dynamic group registry;

loading a client side communications server program in each publication computer in response to the appropriate resource locator therefor for responding to resource locators directed to the client side communications server program and for directing the database management program in organizing the dynamic group registry;

designating a domain computer having electronic storage media for storing a dynamic client registry thereon and for storing resource locators containing function names thereon, the domain computer further comprising a web server program which, when executed by the domain computer, causes the domain computer to respond to the resource locators by calling the function name indicated therein into the domain computer the domain computer further comprising a database management program for organizing the dynamic client registry;

loading a domain communications server program in response to the appropriate resource locator therefore for execution by the domain computer for responding to resource locators directed to the domain communications server program and directing the database management program in organizing the dynamic client registry;

storing a domain communications resource locator list in the domain computer and each publication computer for causing predetermined functions to be selected for execution in the domain communications server in the domain computer;

storing a client side communications resource locator list in the domain computer and each publication computer for causing predetermined functions to be selected for execution in the client side communications server in each publication computer so that communications between the domain computer and each publication computer cause the selected predetermined functions to be executed dynamically in order to manage information communications between the domain computer and each publication computer.

10. The method of claim 9, wherein the step of storing a domain communications resource locator list further comprises the step of storing a register domain communications resource locator for causing the domain communications server program to direct the database management program to register a client side communications server program in the dynamic client registry.

11. The method of claim 9 wherein the step of selecting predetermined functions further comprises the step of selecting one to one communications functions.

12. The method of claim 9 wherein the step of selecting predetermined functions further comprises the step of selecting group to group communications functions.

13. The method of claim 9 wherein the step of selecting predetermined functions further comprises the step of selecting site to site communications functions.

14. The method of claim 9 wherein the step of selecting predetermined functions further comprises the step of selecting publication distribution functions.

15. The method of claim 9 wherein the step of selecting predetermined functions further comprises the step of selecting publication editing functions.

16. The method of claim 9 wherein the step of selecting predetermined functions further comprises the step of selecting publication classification functions.

* * * * *

APPENDIX E

U.S. Patent No. 6,125,388 to Reisman

US006125388A

United States Patent [19]

Reisman

[11] Patent Number: 6,125,388

[45] Date of Patent: Sep. 26, 2000

[54] **SYSTEM FOR TRANSPORTING INFORMATION OBJECTS BETWEEN A USER STATION AND MULTIPLE REMOTE SOURCES BASED UPON USER MODIFIABLE OBJECT MANIFEST STORED IN THE USER STATION**

[76] Inventor: **Richard R. Reisman**, 20 E. 9th St. #14K, New York, N.Y. 10003

[21] Appl. No.: **08/982,157**

[22] Filed: **Dec. 1, 1997**

Related U.S. Application Data

[63] Continuation of application No. 08/251,724, May 31, 1994, Pat. No. 5,694,546.

[51] **Int. Cl.**[7] **G06F 15/16**; G06F 15/163; G06F 15/76; G06F 7/00

[52] **U.S. Cl.** **709/218**; 709/200; 709/201; 709/203; 709/217; 709/219; 709/250; 709/300; 709/303; 707/1; 707/10; 707/100; 707/103; 707/104; 712/28

[58] **Field of Search** 345/333, 331, 345/356; 375/257; 379/45, 112; 395/712, 500.46, 701, 702, 703, 710; 364/134; 370/407, 420; 380/4, 24, 49; 428/142; 705/27, 35, 37; 707/1, 3, 8, 10, 100, 101, 103, 104, 200, 201, 203, 204, 205, 206, 303, 305; 709/101, 104, 201, 203, 206, 217, 218, 221, 223, 224, 231, 236, 246, 250, 300, 303, 219; 711/111, 133, 152, 203; 712/28

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,274,139	6/1981	Hodgkinson et al.	709/203
4,432,057	2/1984	Daniell et al.	707/8
4,528,643	7/1985	Freeny, Jr.	364/900
4,558,413	12/1985	Schmidt et al.	707/203
4,604,686	8/1986	Reiter et al.	395/500.46
4,635,189	1/1987	Kendall	364/200
4,714,992	12/1987	Gladney et al.	707/206
4,714,995	12/1987	Materna et al.	707/201
4,745,559	5/1988	Willis et al.	705/37
4,746,559	5/1988	Nishikawa	428/142
4,760,572	7/1988	Tomikawa	370/94
4,815,030	3/1989	Cross et al.	707/10
4,935,870	6/1990	Burke, Jr. et al.	709/203
4,974,149	11/1990	Valenti	709/217
4,984,155	1/1991	Geier et al.	364/401

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

360229138A	11/1985	Japan	G06F 7/28
362053085A	3/1987	Japan	H04N 7/173
403230234A	10/1991	Japan	G06F 12/00
404142620A	5/1992	Japan	G06F 9/06
WO 97/32251	of 0000	WIPO .	

OTHER PUBLICATIONS

"Head's Up Your Story Request," by Mike Langberg, San Jose Mercury News, CA, CD-ROM Column, Knight Ridder, Tribune Business News, File 10424154.800, Apr. 24, 1994.

"CD-ROM's: They're Not Just for Entertainment," by Laurie Flynn, The New York Times, Sunday, Apr. 24, 1994, p. 10.

1994 Intermediate Conference, San Jose, CA, Mar. 1-3, 1994 (7 pages.).

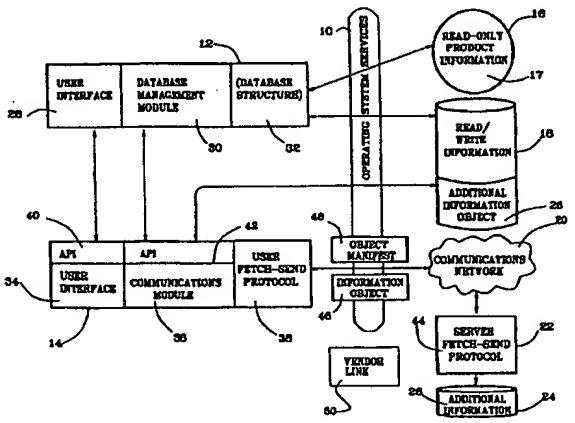
(List continued on next page.)

Primary Examiner—Thomas C. Lee
Assistant Examiner—Tanh Nguyen
Attorney, Agent, or Firm—Westerlund & Powell, P.C.; Robert A. Westerlund; Raymond H.J. Powell Jr.

[57] **ABSTRACT**

A novel electronic information transport component can be incorporated in a wide range of electronic information products, for example magazine collections, to automate the mass distribution of updates, such as current issues, from a remote server to a wide user base having a diversity of computer stations. Advantages of economy, immediacy and ease of use are provided. Extensions of the invention permit automated electronic catalog shopping with order placement and, optionally, order confirmation. A server-based update distribution service is also provided.

30 Claims, 6 Drawing Sheets



U.S. PATENT DOCUMENTS

5,008,814	4/1991	Mathur	709/221
5,008,853	4/1991	Bly et al.	345/331
5,019,963	5/1991	Alderson et al.	707/201
5,133,075	7/1992	Risch	707/201
5,155,847	10/1992	Kirouac et al.	709/221
5,157,783	10/1992	Anderson et al.	395/600
5,185,697	2/1993	Jacobs et al.	379/45
5,187,787	2/1993	Skeen et al.	709/300
5,220,501	6/1993	Lawlor et al.	380/24
5,220,657	6/1993	Bly et al.	711/152
5,226,161	7/1993	Khoyi et al.	709/303
5,257,369	10/1993	Skeen et al.	709/300
5,287,504	2/1994	Carpenter et al.	707/201
5,297,249	3/1994	Bernstein et al.	345/356
5,303,379	4/1994	Khoyi et al.	395/710
5,319,542	6/1994	King, Jr. et al.	364/401
5,359,730	10/1994	Marron	395/712
5,379,424	1/1995	Morimoto et al.	395/600
5,404,488	4/1995	Kerrigan et al.	711/133
5,406,557	4/1995	Baudoin	370/407
5,421,009	5/1995	Platt	709/221
5,426,747	6/1995	Weinreb et al.	711/203
5,440,744	8/1995	Jacobson et al.	709/203
5,442,791	8/1995	Wrabetz et al.	709/104
5,452,447	9/1995	Nelson et al.	707/205
5,473,772	12/1995	Halliwell et al.	395/712
5,485,370	1/1996	Moss et al.	709/217
5,491,800	2/1996	Golsmith et al.	709/221
5,491,820	2/1996	Belove et al.	707/3
5,495,610	2/1996	Shing et al.	709/221
5,497,491	3/1996	Mitchell et al.	709/303
5,499,343	3/1996	Pettus	709/203
5,515,508	5/1996	Pettus et al.	709/203
5,519,769	5/1996	Weinberger et al.	379/112
5,519,875	5/1996	Yokoyama et al.	709/303
5,528,490	6/1996	Hill	395/712
5,530,852	6/1996	Meske	709/206
5,539,735	7/1996	Moskowitz	370/420
5,541,991	7/1996	Benson et al.	380/4
5,544,320	8/1996	Konrad	79/203
5,548,726	8/1996	Pettus	709/221
5,553,223	9/1996	Greenlee et al.	345/333
5,555,427	9/1996	Aoe et al.	709/201
5,557,793	9/1996	Koerber	707/103
5,557,798	9/1996	Skeen et al.	705/35
5,560,012	9/1996	Ryu et al.	395/701
5,564,051	10/1996	Halliwell et al.	707/200
5,566,302	10/1996	Khalidi et al.	709/300
5,572,643	11/1996	Judson	709/218
5,577,244	11/1996	Killebrew et al.	395/703
5,577,251	11/1996	Hamilton et al.	709/101
5,581,755	12/1996	Koerber et al.	707/103
5,581,761	12/1996	Radia et al.	395/702
5,581,764	12/1996	Fitzgerald et al.	395/703
5,586,311	12/1996	Davies et al.	707/1
5,586,326	12/1996	Ryu et al.	395/701
5,594,910	1/1997	Filippe et al.	712/28
5,596,720	1/1997	Hamada et al.	709/206
5,596,746	1/1997	Shen et al.	707/101
5,600,834	2/1997	Howard	707/201
5,602,993	2/1997	Stromberg	395/712
5,606,493	2/1997	Duscher et al.	364/134
5,608,874	3/1997	Ogawa et al.	709/246
5,615,112	3/1997	Liu Sheng et al.	707/104
5,619,710	4/1997	Travis, Jr. et al.	709/203
5,623,605	4/1997	Keshav et al.	709/236
5,623,656	4/1997	Lyons	707/10
5,623,661	4/1997	Hon	707/1
5,625,818	4/1997	Zarmer et al.	707/104
5,628,005	5/1997	Hurvig	707/8
5,630,066	5/1997	Gosling	709/221
5,630,092	5/1997	Carreiro et al.	711/111
5,630,116	5/1997	Takaya et al.	707/201
5,634,010	5/1997	Ciscon et al.	709/223
5,640,564	6/1997	Hamilton et al.	709/303
5,644,764	7/1997	Johnson et al.	707/103
5,646,992	7/1997	Subler et al.	380/4
5,649,192	7/1997	Stucky	707/103
5,652,887	7/1997	Dewey et al.	709/303
5,664,207	9/1997	Crumpler et al.	707/505
5,668,997	9/1997	Lynch-Freshner et al.	707/303
5,673,322	9/1997	Pepe et al.	380/49
5,680,548	10/1997	Trugman	709/226
5,680,617	10/1997	Gough et al.	707/104
5,682,532	10/1997	Remington et al.	709/303
5,684,984	11/1997	Jones et al.	707/10
5,684,991	11/1997	Malcolm	707/204
5,689,708	11/1997	Regnier et al.	709/229
5,694,549	12/1997	Carlin et al.	709/250
5,706,434	1/1998	Kremen et al.	709/218
5,708,780	1/1998	Levergood et al.	395/200.12
5,710,918	1/1998	Lagarde et al.	707/10
5,715,314	2/1998	Payne et al.	380/24
5,721,911	2/1998	Ha et al.	707/100
5,724,424	3/1998	Gifford	380/24
5,761,649	6/1998	Hill	705/27
5,761,677	6/1998	Senator et al.	707/203
5,761,678	6/1998	Bendert et al.	707/204
5,768,528	6/1998	Stumm	709/231
5,790,793	8/1998	Higely	709/218
5,809,076	9/1998	Hofmann	375/257
5,812,776	9/1998	Gifford	395/200.47
5,862,325	1/1999	Reed et al.	709/201
5,909,492	6/1999	Payne et al.	380/24

OTHER PUBLICATIONS

"Microsoft Complete Baseball" Product Brochure, Microsoft Corp. product announced Mar. 1, 1994.

CompuServe Introductory Membership, CompuServe, Print Date Dec. 1992.

"Plug and Play Making Add–In Cards Play Automatically," Intel Technology Briefing (4 pages).

Daily Federal Register, 1993, Counterpoint Publishing, Fall 1993.

The Federal Register, Product Brochure, Counterpoint Publishing, Fall 1993.

"Counterpoint's Compact Disk Federal Register" order form, Counterpoint Publishing.

"Code of Federal Regulations on CD–ROM," Counterpoint Publishing.

"Compact Disk of Federal Register" price list, Counterpoint Publishing Co., Jul. 1, 1993.

"Microsoft Messaging Application Program Interface (MAPI)," created Jan. 1993, Microsft Corp.

"WOSA Backgrounder: Delivering Enterprise Services to the Windows–Based Desktop," created Jul. 1993, Microsoft Corp.

"RemoteWare Software Licenses," fee list, Xcellenet, Inc., Aug. 16, 1993.

"RemoteWare Server," product brochure, Xcellenet, Inc., 1992.

"RemoteWare Communications Management System," product brochure, Xcellenet, Inc.

"RemoteWare Mail," product brochure, Xcellenet, Inc.

"RemoteWare Documents," product brochure, Xcellenet, Inc., 1992.

"Remote Ware Reports," product brochure Xcellenet, Inc., 1992.

"Software Solution Helps Pull Satellite Offices into Network Environment," by Kathleen Doler, Investors Business Daily, Mar. 10, 1993, vol. 9, No. 231.
"Simple MAPI," Microsoft Corp., 1993.
The Frye Utilities for Networks, "Software Update and Distribution System," Frye Computer Systems, Inc., Copyright 1992.
David Snyder, "The Poor Man's Mirror Script," 8 page printout of software documentation, Nov. 30, 1984.
Fitzpatrick et al., "Automatic Mirroring of the IRAF FTP and WWW Archives," Contents of WWW Web Site, http//iraf.nano.edu:80/project/mirror/, as of Mar. 18, 1997.
Online Business Today Archives, Home Page Press, Inc., Contents of WWW Web Site, http://www.hpp.com/s–clickshare95.html, as of Mar. 19, 1997.
"Go–Get–It, Internet Personal Agent Thrills Net Users," NorthTech Software Inc., 1994, Contents of WWW Web Site, http://www.hpp.com/gogetit.html, as of Mar. 19, 1997.
"Mirror Applescript—Find Your Salvation with a Mirror Script for the Macintosh?," Jim Matthews, Contents of http://www.darmouth.edu/pages/softdev/fetch.html, as of Mar. 18, 1997.
"NetTerm—The Ultimate telnet experience?," InterSoft International, Inc., Contents of http://starbase.neosoft.com/zkrr01/netterm.html, as of Mar. 18, 1997.
"Intermind Announces Approval of First Patent Application," Intermind Corporation Press Release, http://www.intermind.com/inside/press_rel/100797/allow.html, as of Oct. 1997.
"About Intermind's Channel Communications Patents," Intermind's Patent Description, http://www.intermind.com/material/patent desc.htm.l.
"Frequently Asked Questions about Intermind's Patents," Intermind Corporation Patent FAQ, http://www.intermind-.com/material/patent_faq.html.
"ProComm" Reference Manual, Datastorm Technologies, Inc., 1986.
"Central Point Commute: Fast Remote Control of PCS Running Windows or DOS," Central Point Sortware, Inc., 1992.
"Beyond Store and Forward: Extending ESD: Xcellenet Routing Server Cut Out the Middleman," Data Communications, Jun. 1993.
"CompuServe" Quick Start Guide, CompuServe, 1994.
"When Worlds Collide," CD Rom World, Nov. 1994.
Intermedia Conference, (transcript), Mar. 1994., pp. 81–83 and 1 other.
"Exploring Hybrid World of CD–Rom/On–Line Products," Multimedia Week, Mar. 7, 1994.
"What is CompuServeCD?," received Apr. 29, 1994.
"Software Distribution Is Still a Bad Dream," PC Week, Mar. 28, 1994, p. 54.
"WebWhacker," (Guide), ForeFront Group, Inc., 1995.
"Browsers Make Navigating the World Wide Web a Snap," The New York Times, Jan. 29, 1995.
"Interactive Media Woks Debuts sampleNET," Interactive Media Works, news release dated Jul. 17, 1995.
Digital Delivery Product Data Sheet, Digital Delivery, Inc., Sep. 17, 1995.
Digital Delivery Launches Unique Delivery Agent; Introduces First Delivery Agent to Seamlessly Provide PC Users With Fully–Formatted Content, Business Wire, Oct. 3, 1995.
CMP is First Technology Publisher to Deliver Web Content to the Desktop, CMP News, Oct. 18, 1995.
"The Future of Audio CD's," Microsoft CD Plus Event Page, Microsoft Corp., Dec. 4, 1995.
"Marketing via Teleshuttle," Letter from Richard Reisman to Lou Jordan dated May 22, 1995 with enclosure,"A Short White Paper on the Teleshuttle Solution," Teleshuttle, May 5, 1995.
"LinkStar Announces Site Launcher," LinkStar Communications Corporation, Feb. 26, 1996.
"Frontier Technologies' CyberSearch 2.0 Internet Search Tool Now Works With Popular Browsers," Frontier Technologies Corporation, Apr. 5, 1996.
"CyberSearch 2.0 Beta Manual," Frontier Technologies Corporation, Apr. 5, 1996.
"CD Plus Technical Information," Microsoft CD Plus Event Page, Microsoft Corp., Dec. 4, 1995.
"Open Market Announces New Desktop Software to Delivery Resources of the World Wide Web Directly to the User," OM–Express News Release, Apr. 16, 1996.
"How to Publish an OM–Express Package," OM–Express Product Information, Apr. 16, 1996.
"MarketScape WebCD 1.0 Bypasses Internet Bottlenecks," MarketScape Press Release, MarketScape, Inc., Aug. 26, 1996.
"Reality's Wealth Builder 3.0" User's Guide, Money Magazine, Reality Technologies, Inc., 1992.
"Reality's Wealth Builder" Version 3.1 Supplement, Money Magazine, Reality Technologies, Inc., Apr., 1993.
"Aren't you glad you waited?," QmodemPro for Windows Advertisement, Mustang Software, Inc.
"XcelleNet RemoteWare: Integrated Mobile Communications," Operations Automation Strategies Research Note, Jul. 26, 1993.
"About Marimba: A Word from Marimba's President and CEO, Kim Polese," Contents of WWW Web Site http://www.marimba.com/about executive_over.html as of Nov. 17, 1998.
"Marimba Products: Marimba's Castanet™: An Essential Part of Your E–Business InFrastructure," Content of WWW Web Site http://www.marimba.com/product/content/product.html as of Nov. 17, 1998.
"Marimba Library," Contents of WWW Web Site, http://www.marimba.com/datasheets/ as of Nov. 17, 1998.
Dow Jones News Service (DowVision™) Lecture Presentation Handout, lecture presented by Charles I. Brady at the Wall Street Workstation Conference, New York City, NY, Oct. 11–12, 1989.
Lecture Presentation Notes on the Folio World–Wide–Web Retriever 3.1, presented by Jeff Gammon at the Infobase '95 Conference, Copyright 1995.
C. Bowman, P. Danzig, D. Hardy, U. Manber, M. Schwartz & D. Wessels "Harvest: A Scalable, Customizable Discovery and Access System" Mar. 12, 1995.
D. Hardy & M. Schwartz "Customized Information Extraction as a Basis for Resource Discovery" Mar. 1994.
William G. Camargo "The Harveset Broker," Dec. 1994.
D. Bulterman, G. van Rossum and R. van Liere "A Structure for Transportable, Dynamic Multimedia Documents" US–ENIX, Summer '91 Nashville, TN.
G. Almes and C. Holman "Edmas: An Object–Oriented, Locally Distributed Mail System" IEEE Transactions on Software Engineerng, Sep. 1987.
G. Almes, A. Black, C. Bunie and Weibe "Edmas: A Locally Distributed Mail System" IEEE, 1984.

W. Bender, H. Lie, J. Orwant, L. Teodosio, & N. Abramsom "Newspace: Mass Media and Personal Computing," USENIX—Summer '91—Nashville TN.

R. Thomas, H. Forsdick, T. Crowley, R. Schaaff, R. Tomlinson & V. Travers "Diamond: A Multimedia Message System Built on a Distributed Architecture" IEEE, Dec. 1994.

S. Ramanathan & P.V. Rangan "Architectures for Personalized Multimedia" IEEE, 1994.

N. Yankelovich, B. Haan, N. Meyrowitz & S. Drucker "Intermedia: The Concept and the Construction of a Seamless Information Environment" IEEE, Jan. 1988.

D. Woelk, W. Kim & W. Luther, "An Object–Oriented Approach to MultiMedia Database," ACM 1986.

N. Borenstein. C. Everhart, J. Rosenberg, A Stoller "A Multi–media Message System for Andrew" USENIX Winter Conference Feb., 1988.

S. Jackson & N. Yankelovich "InterMail: A Prototype Hypermedia Mail System" Hypertext 91 Proceedings Dec. 1991.

E. Hoffert & G. Gretsch, "The Digital News System at Ed.ucom: A Convergence of Interactive Computing Newspaper, Television and High Speed Networks" Communications of the ACM Apr. 1991.

D. Crocker, E. Szurkowski & D. Farber "An Internetwork Memo Distribution Capability—MMDF" IEEE, ACM 1979.

Douglas Engelbart "Authorship Provisions in Augment" IEEEE, 1984.

J.J. Garcia–Luna–Aceves "Towards Computer–Based Multimedia Information Systems" Computer Message System 85, 1986.

Debra P. Deutsch, "Implementing Distribution Lists in Computer–Based Message Systems," Computer–Based Message Services, IFIP, 1984.

T. Purdy, D. Thorslund & N. Witchlow "Meridian SL Messaging" Computer Message Systems—85 IFIP, 1986.

Michael Tschichholz "Message Handling System: Requirements to the User Agent" Computer Message Systems—85, IFIP, 1986.

Lother Wosnitza "Group Communication in the MHS Context" Computer Message Systems 85 IFIP, 1986.

Jacob Palme "Distribution Agents (mailing lists) in Message Handling Systems" Computer Message Systems 85 IFIP, 1986.

Teresa F. Lunt "A Model for Message System Security" Computer Message Systems 85 IFIP, 1986.

A. Roger Kaye "A User Agent for Multiple Computer–Based Message Services" Computer–Based Message Services, IFIP 1984.

Paul Wilson "Structure for Mailbox System Applciations" Computer–Based Message Services, IFIP 1984.

J. Postel, G. Finn, A. Katz & J. Reynolds "The ISI Experimental Multimedia Mail System" Information Sciences Institute, Sep. 1986.

E. Moeller, A Scheller & G. Schurmann "Distributed Processing of Multimedia Information" IEEE Computer Society Proceedings May 28–Jun. 1, 1990.

Richard L. Phillips "An Interpersonal Multimedia Visualization System" IEEE Computer Graphics & Applications IEEE 1991.

Jacob Palme "You Have 134 Unread Mail! Do You Want to Read Them Now?" Computer–Based Message Services IFIP, 1984.

M. Papa, G. Raguiccini, G. Corrente, M. Ferrise, S. Giurleo and D. Vitale "The Development of an Object–Oriented Multimedia Information System" Lecture Notes in Computer Science, Sep. 1994.

Silvano Maffeis "A Flexible System Design to Support Object–Groups and Object–Oriented Distributed Programming" Lecture Notes in Computer Science, Jul. 1993.

R. Gotze, H. Eirund & R. Claass in "Object–Oriented Dialog Control for Multimedia User Interfaces" Lecture Notes in Computer Science—Human Computer Interaction Sep. 1993.

Chris Maeda "A Metaobject Protocol for Controlling File Cache Management" Lecture Notes in Computer Science, Mar. 1996.

A. Joseph, A. deLespinasse, J. Tauber, D. Gifford & M. Kaashoek "Rover" A Toolkit for Mobile Information Access SIGOPS '95 1995. ACM.

Wolfgang Lux "Adaptable Object Migration: Concept and Implementation" Operating Systems Review Apr. 1995.

R. Campbell, N. Islam, R. Johnson, P. Kougiouris & P. Madany "Choices, Frameworks and Refinement" Department of Computer Science, University of Illinois, Dec. 1991.

Klemens Bohm & Thomas C. Rakow "Metadata for Multimedia Documents" SIGMOD Record, vol. 23, No. 4, Dec. 1994.

Simon Gibbs "Composite Multimedia and Active Objects" OOPSLA '91.

T. Purdin, R. Schlichting & G. Andrews "A File Replication Facility for Berkeley Unix" Software Practice and Experience, vol. 17, Dec. 1987.

A. Black, N. Hutchinson, E. Jul & H. Levy "Object Structure in the Emerald System" OOPSLA '86 Proceedings.

Daniel T. Chang "Coral: A Concurrent Object–Oriented System for Constructing and Executing Sequential, Parallel and Distributed Applications" OOPS Messenger, Apr. 1991.

A. Birrell, G. Nelson, S. Owicki & E. Wobber "Network Objects" Proceedings of the 14th ACM Symposium on Operating Systems Principles, Dec. 5–8, 1993.

Jacques Ferber "Computational Reflection in Class based Object Oriented Languages" OOPSLA '89 Proceedings.

Michael Caplinger "An Information System Based on Distributed Objects" OOPSLA '87 Proceedings.

C. Fung & M. Pong "MOCS: an Object–Oriented Programming Model for Multimedia Object Communication and Synchronization" 1994 IEEE.

T. Hase & M. Matsuda "A New Audio–Visual Control Using Message Object Transmission", 1994 IEEE, Nov. 1994.

F. Horn & J. Stefani "On Programming and Supporting Multimedia Object Synchronization" The Computer Journal, vol. 36, No. 1, 1993.

T. Little & A. Ghafoor Spatio–Temporal Composition of Distributed Multimedia Objects for Value–Added Networks, IEEE, 1991.

M. Vazirgiannis & C. Mourlas "An Object–Oriented Model for Interactive Multimedia Presentations" The Computer Journal, vol. 36, No. 1, 1993.

T. Little & A. Ghafoor "Synchronization and Storage Models for Multimedia Objects" 1990 IEEE, Apr. 1990.

Cosmos Nicolaou "Architecture for Real–Time Multimedia Communications Systems", 1990 IEEE, Apr. 1990.

Ralf Steinmetz "Synchronization Properties in Multimedia Systems" 1990 IEEE, Apr. 1990.

T. Little & A. Ghafoor "Network Considerations for Distributed Multimedia Object Composition and Communication" 1990 IEEE Network Magazine, Nov. 1990.
K. Smith and S. Zdonik "Intermedia: A Case Study of the Differences Between Relational and Object–Oriented Database Systems" OOPSLA '87 Proceedings.
S. Ramanathan & P. Rangan "Architectures for Personalized Multimedia" 1994 IEEE.
Marvin Sirbu and J.D. Tygar, "Netbill: An Intenet Commerce System Optimized For Network–Delivered Services", IEEE Personal Communications Magazine, pp. 34–39, Aug. 1995.
Henrik Eriksson, "Expert System As Knowledge Servers", IEEE Expert Magazine, pp. 14–19, Jun. 1996.
Budi Yuwono and Dik Lun Lee, "Wise: A World Wide Web Resource Database System", IEEE Transactions on Knowledge and Data Engineering, vol. 8, No. Aug. 1996.
H. Penny Nii "Blackboard Systems" The A1 Magazine, Summer, 1986.
AppleShare, Apr. 1995.
"Manual Page for Unix NFS Mount Command".
"Manual Page for Unix FSTAB Command".
Phil Lapsley and Brian Kantor "Network News Transfer Protocol", Feb. 1986.
Brian Kantor and Phil Lapsley, Network News Transfer Protocol, "A Proposed Standard for the Stream–Based Transmission of News", Feb. 1986.
M. Crispin "Network Working Group", University of Washington, Dec. 1996.
Terry Gray Comparing Two Approaches to Remote Mailbox Access: IMAP vs. POP, University of Washington.
Terry Garry "Message Access Paradigms and Protocols", University of Washington, Aug. 1995.
UNIX User's Reference Manual (URM), 4.3 Berkley Software Distribution Virtual VAX–11 Version, Apr. 1986 (including the description of a RDIST—a remote file distribution program—4 pages).
UNIX System Manager's Manual (SMM), 4.3, Berkeley Software Distribution Virtual VAX–11 Version, Apr. 1986 (including a document entitled "A Fast File System for UNIX," by Marshall McKusick et al., pp. SMM 14–1 through SMM 14–15).
Nachbar, Daniel, "When Network File Systems Aren't Enough: Automatic Software Distribution Revisited," Proceedings of the USENIX Association Summer Conference, pp. 159–171, Atlanta, GA, Jun. 9–13, 1986 (including a description of a method and system for updating computer files called "Track").
Mockapetris, P., "Domain Names—Implementation and Specification," Network Working Group, Request for Comments—883, Nov. 1983.
Saltzer, J., "On the Naming and Binding of Network Destinations," Network Working Group, Request for Comments—1498, Aug. 1993.
A. DeScon and R. Braden, Background File Transfer Program (BFTP), Network Working Group Request for Comments: 1068, Aug. 1988.
Lee McLoughlin, usr/bin/perl—Mirror Master—Run Several Mirrors in Parallel, URL:strucbio.biologie.uni–konstanz.de/pdb/mirror/mm, Jan. 18, 1994.
Jack Lund, local/bin/perlbin/perl—urlget—Get a Document Given a WWW URL, URL:www.chemie.uni–dortmund.de/–loki/exp/urlget, Mar. 23, 1994.
Mirror(1L) Misc. Reference Manual Pages Mirror(1L)Name mirror—mirror packages on remote sites, URL:nic.funet.fi/FUNET/hamster/mirror.txt, Dec. 2, 1993.
Article 5397 of comp.lang.perl: Xref: feenix.metronet.com comp.infosystems.www:1336 comp.lang.perl:5397, URL:ftp.telecom.sk/pub/mirror/CPAN/scripts/i...WWW//http.get.pl, Aug. 25, 1993.
Article 3893 of comp.lang.perl: Xref: feenix.metronet.com comp.lang.perl:3893, URL:www.metronet.com/perlinfo/scripts/ftpstuff/ftpget, Jul. 1, 1993.
Article 10383 of comp.lang.perl., URL:www.metronet.com/perl/scripts/ftpstuff/ftpr, Apr. 6, 1993.

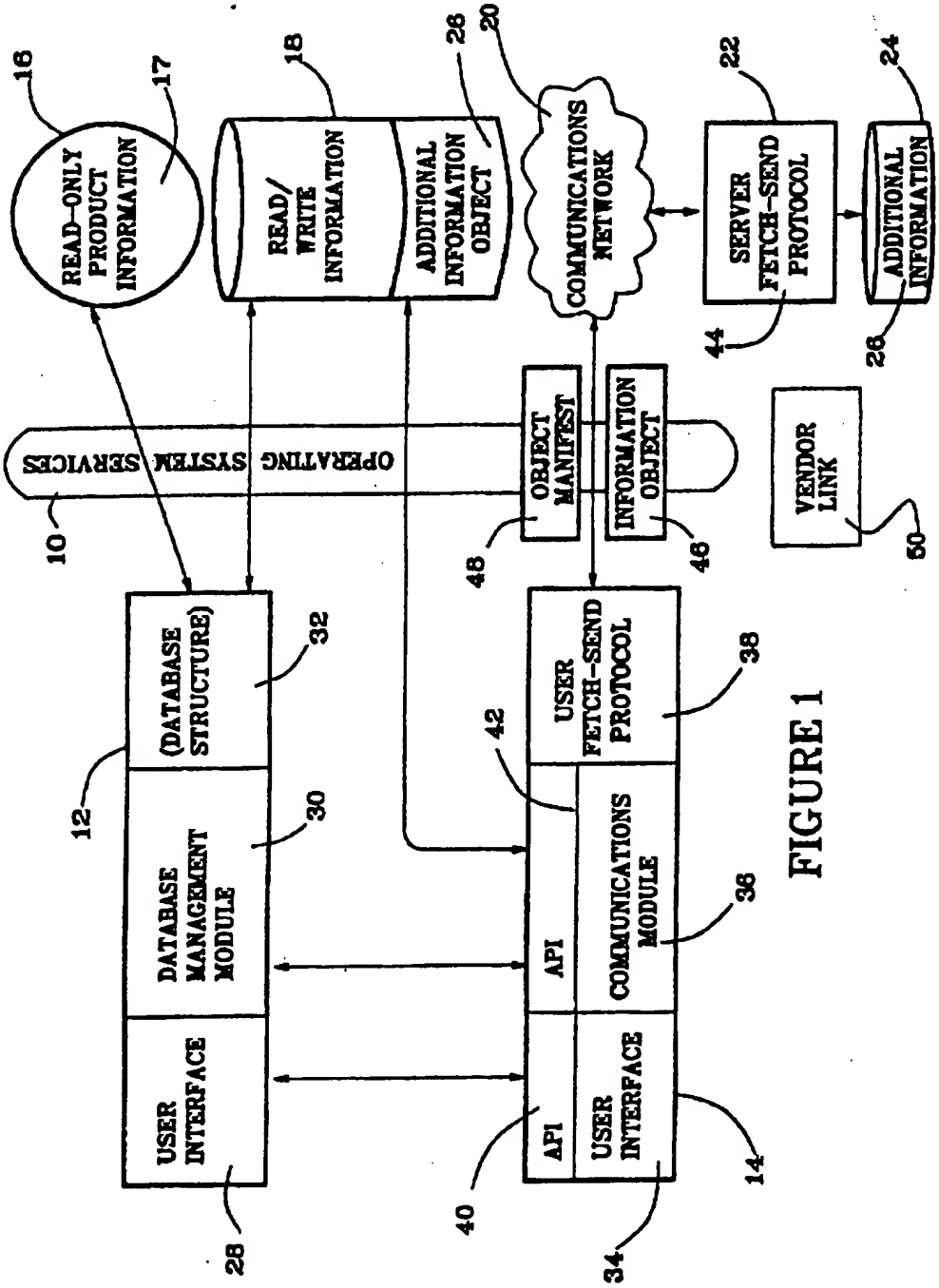

READ-ONLY PRODUCT INFORMATION
16
17
READ/ WRITE INFORMATION
18
ADDITIONAL INFORMATION OBJECT
26
20
COMMUNICATIONS NETWORK
SERVER FETCH-SEND PROTOCOL
22
ADDITIONAL INFORMATION
24
44
28
OPERATING SYSTEM SERVICES
10
OBJECT MANIFEST
INFORMATION OBJECT
48
48
VENDOR LINK
50
(DATABASE STRUCTURE)
32
12
DATABASE MANAGEMENT MODULE
30
USER INTERFACE
28
USER FETCH-SEND PROTOCOL
38
42
API
COMMUNICATIONS MODULE
36
API
USER INTERFACE
40
34
14

FIGURE 1

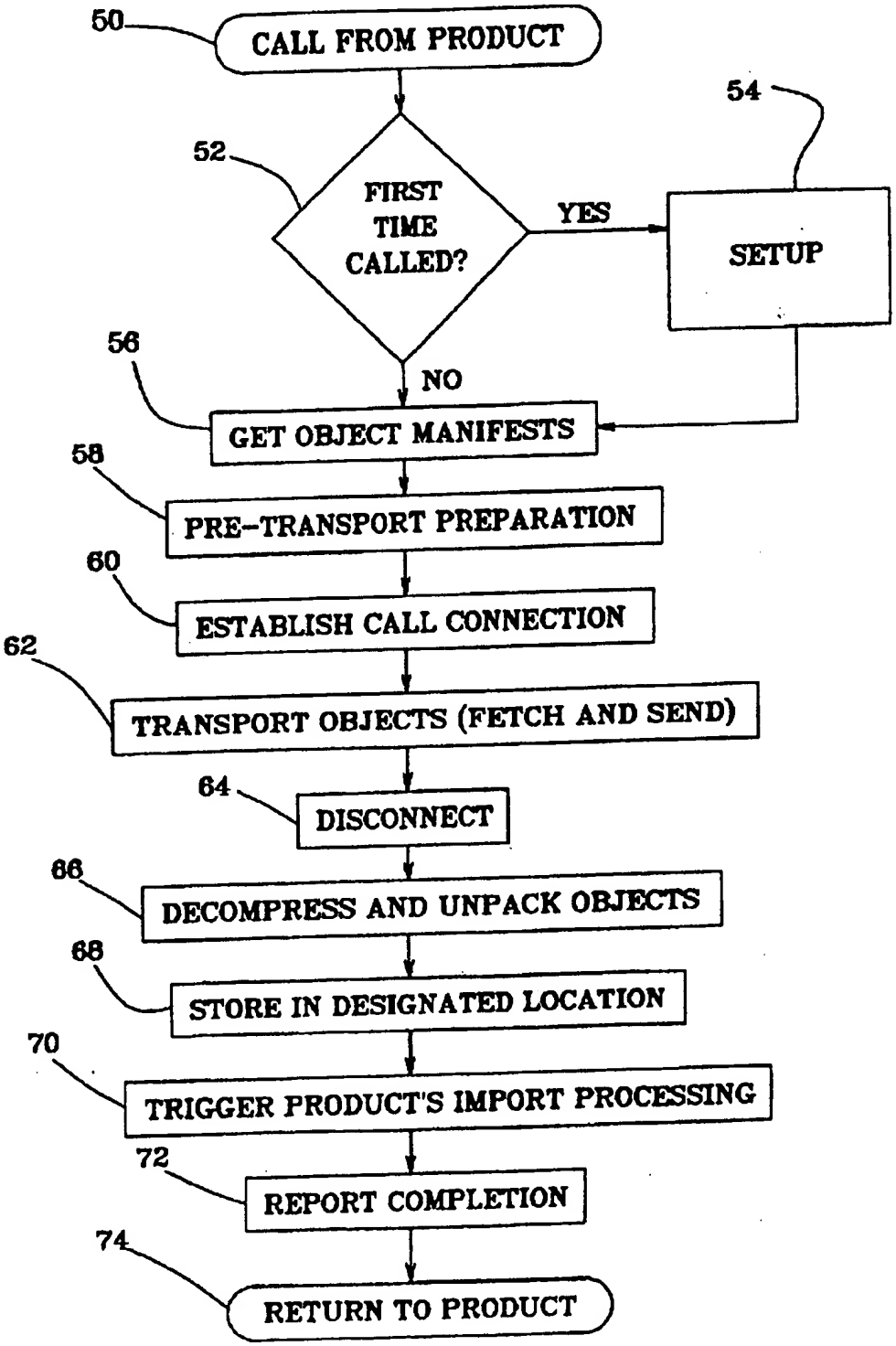

50
CALL FROM PRODUCT
52
FIRST TIME CALLED?
YES
54
SETUP
NO
56
GET OBJECT MANIFESTS
58
PRE-TRANSPORT PREPARATION
60
ESTABLISH CALL CONNECTION
62
TRANSPORT OBJECTS (FETCH AND SEND)
64
DISCONNECT
66
DECOMPRESS AND UNPACK OBJECTS
68
STORE IN DESIGNATED LOCATION
70
TRIGGER PRODUCT'S IMPORT PROCESSING
72
REPORT COMPLETION
74
RETURN TO PRODUCT

FIGURE 2

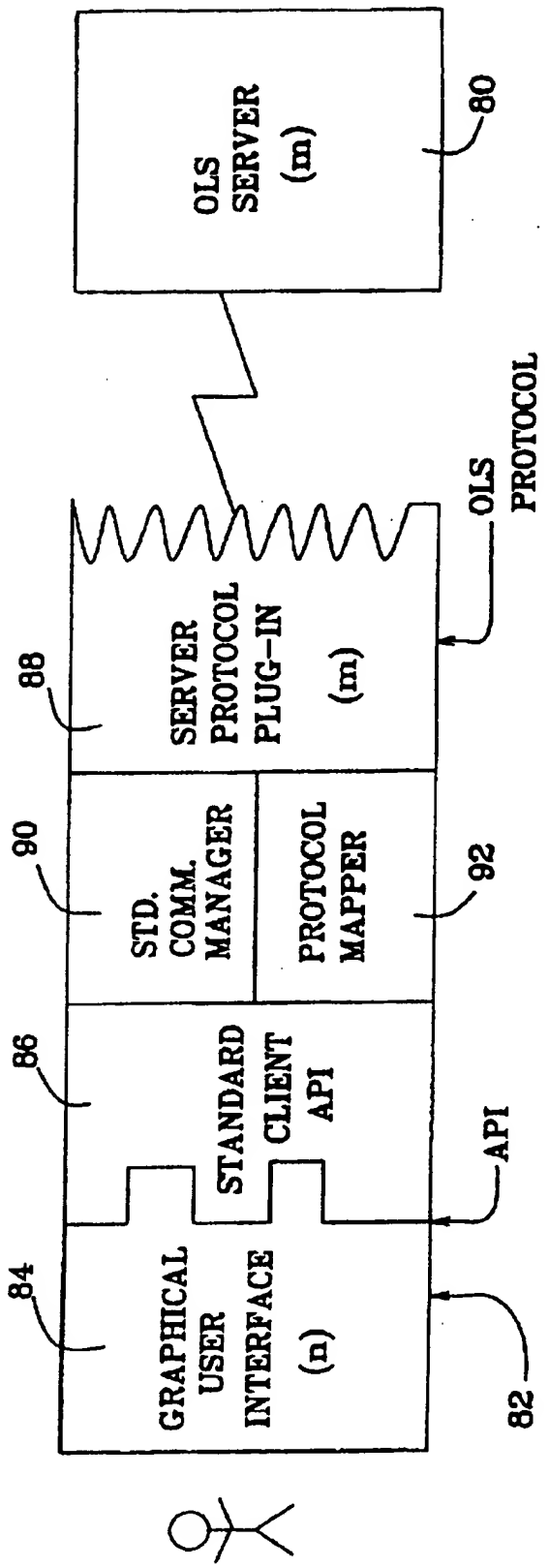

OLS SERVER (m)
80
OLS PROTOCOL
SERVER PROTOCOL PLUG-IN (m)
88
STD. COMM. MANAGER
90
PROTOCOL MAPPER
92
STANDARD CLIENT API
86
API
GRAPHICAL USER INTERFACE (n)
84
82

FIGURE 3

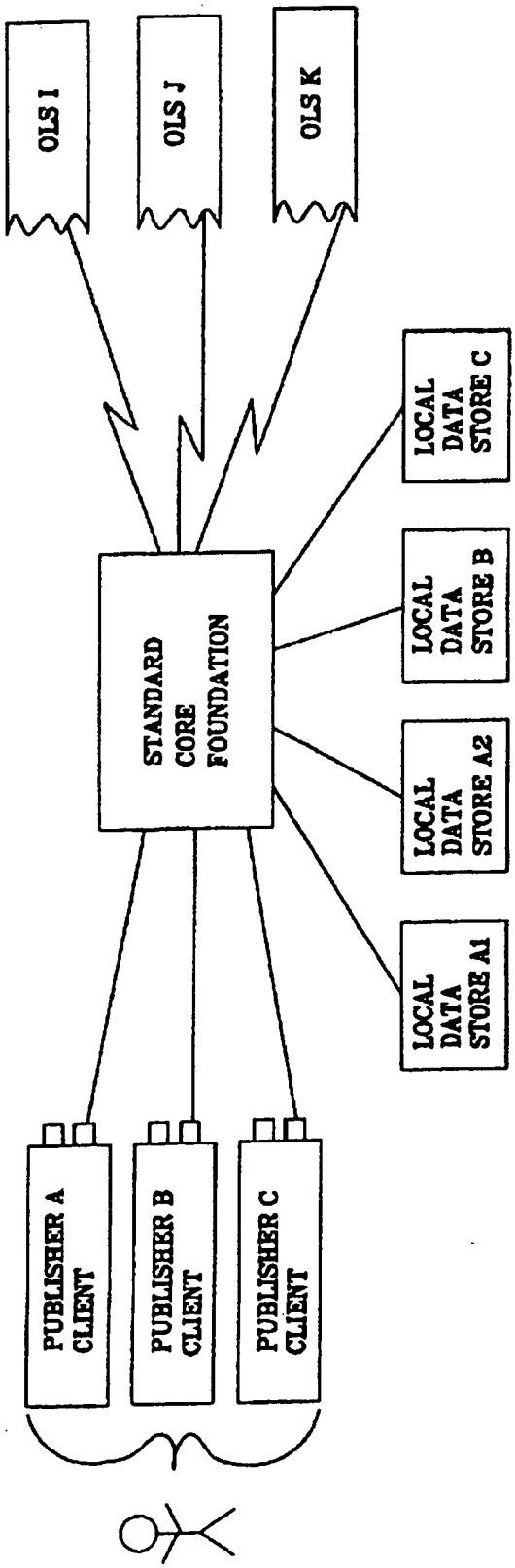
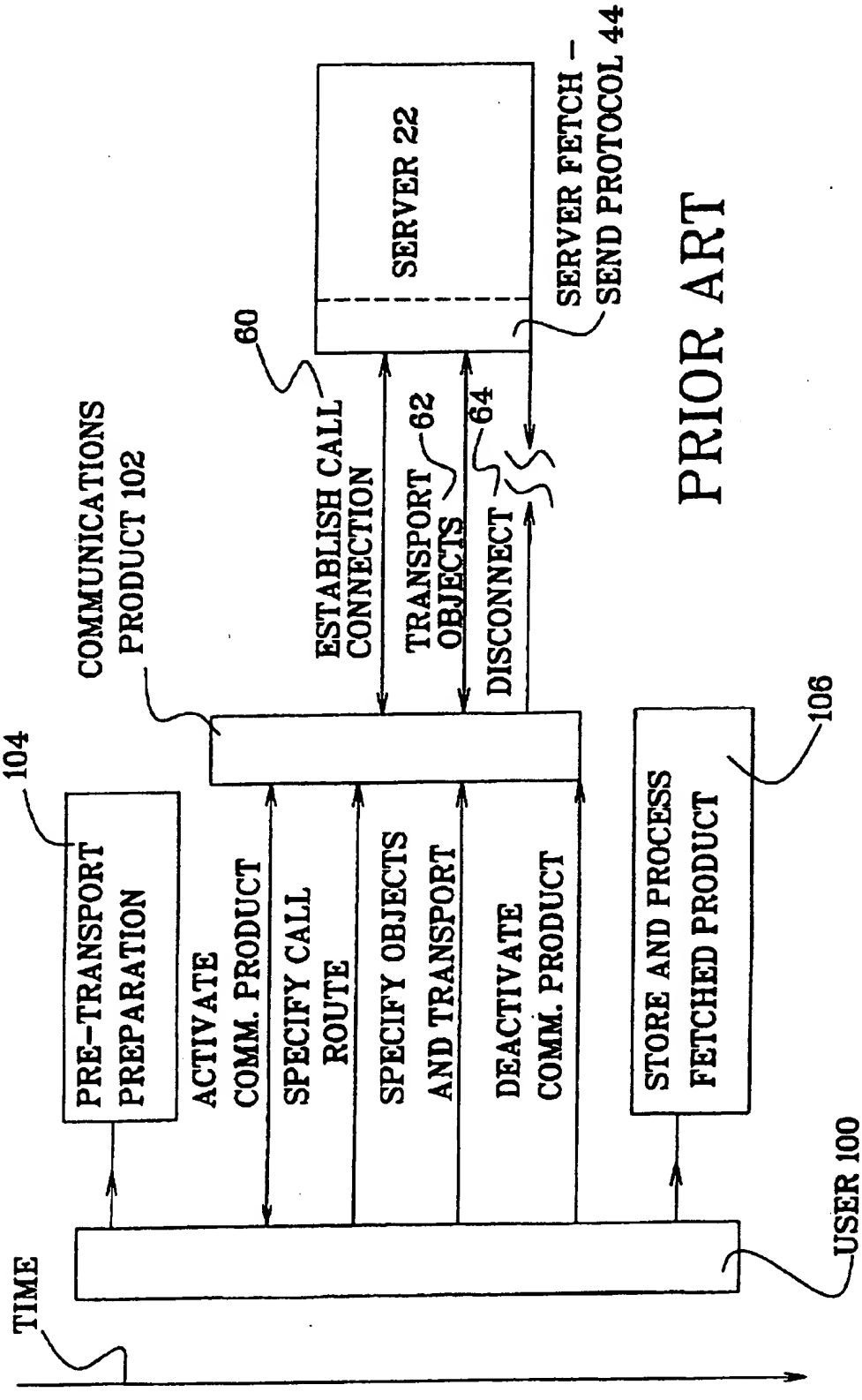

OLS I
OLS J
OLS K
LOCAL DATA STORE C
LOCAL DATA STORE B
LOCAL DATA STORE A2
LOCAL DATA STORE A1
STANDARD CORE FOUNDATION
PUBLISHER A CLIENT
PUBLISHER B CLIENT
PUBLISHER C CLIENT

FIGURE 4



PRIOR ART

FIGURE 5

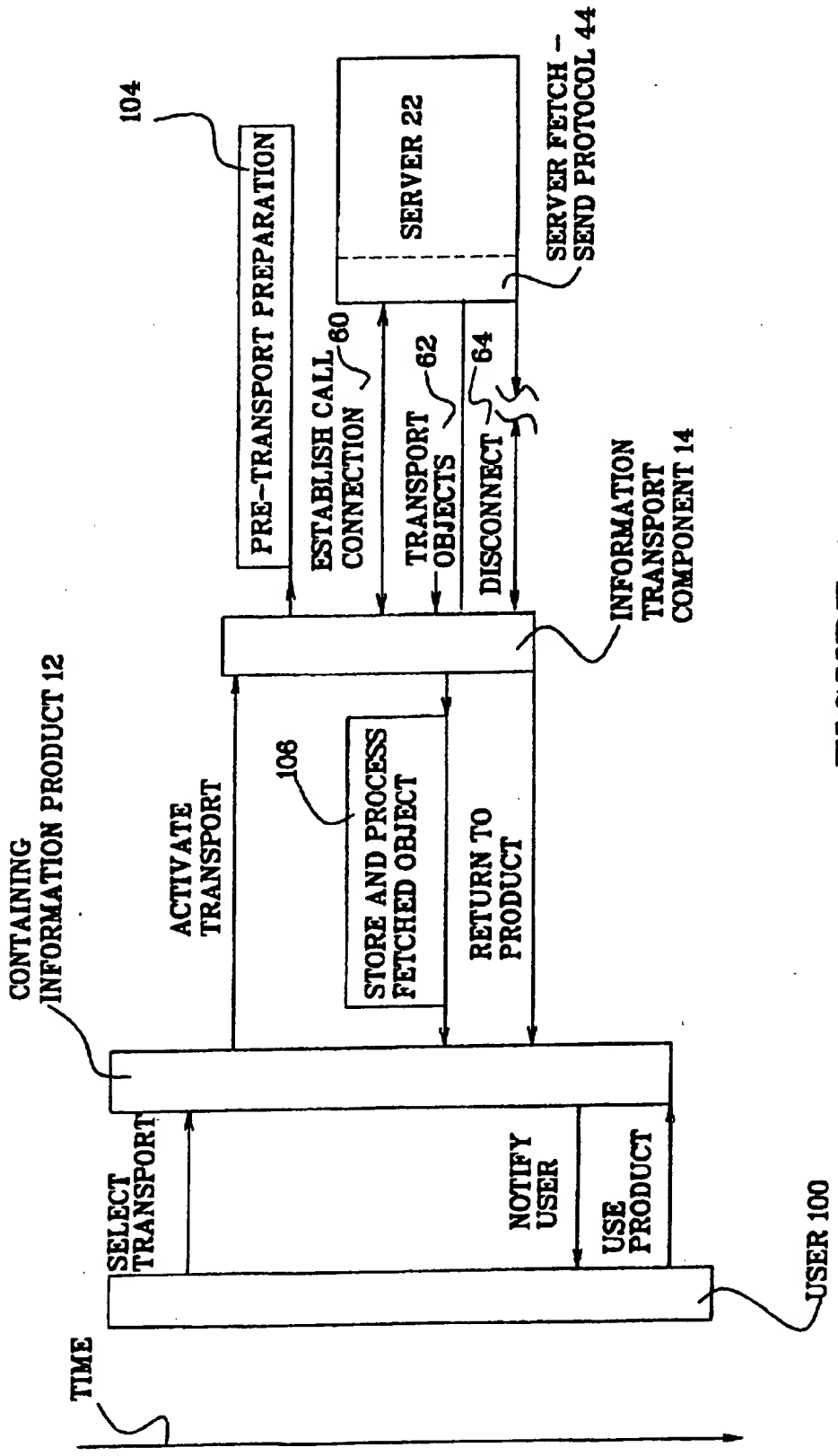

104
PRE-TRANSPORT PREPARATION
SERVER 22
SERVER FETCH - SEND PROTOCOL 44
ESTABLISH CALL CONNECTION 60
TRANSPORT OBJECTS 62
DISCONNECT 64
INFORMATION TRANSPORT COMPONENT 14
CONTAINING INFORMATION PRODUCT 12
ACTIVATE TRANSPORT
106
STORE AND PROCESS FETCHED OBJECT
RETURN TO PRODUCT
SELECT TRANSPORT
NOTIFY USER
USE PRODUCT
USER 100
TIME

FIGURE 6

SYSTEM FOR TRANSPORTING INFORMATION OBJECTS BETWEEN A USER STATION AND MULTIPLE REMOTE SOURCES BASED UPON USER MODIFIABLE OBJECT MANIFEST STORED IN THE USER STATION

This application is a continuation of application Ser. No. 08/251,724 filed May 31, 1994, now U.S. Pat. No. 5,694,546 dated Dec. 2, 1997.

TECHNICAL FIELD

The present invention relates to computer-implemented transport of electronic information objects. More specifically it relates to information transport software which can be used for transporting information objects between a remote server and any one of multiple, uncoordinated intelligent computer workstations. Still more particularly, it provides a computer-implemented software component that can be used to facilitate the distribution of information objects from a remote source to a large number of customers or subscribers.

BACKGROUND

Electronic publication is an exploding industry in which thousands of new products including magazines and periodicals, software applications and utilities, video games, business, legal and financial information and databases, encyclopedias and dictionaries are purchased by millions of customers. Commonly, such information products are replicated in computer-readable form on magnetic or optical storage diskettes and are box-packaged with printed manuals for distribution to retail stores and direct mail sales. These marketing practices are relatively expensive and involve a significant time lag of at least days or weeks to get a product into a consumer's hands once it is created.

Such costs and delays are generally acceptable for original, high value products such as collections of publications or software application, of which some examples are NEWSWEEK® Interactive CD-ROM, or disks, which provides a searchable audio-visual library of issues of NEWSWEEK magazine and CINEMANIA® CD-ROM which provides reviews and other information on newly released films. For time-sensitive, low-value updates, for example, the latest issue of Newsweek or last week's movie reviews, distribution in stored form, on physical media, is slow and the cost may exceed the value of the information in the product.

Thus, electronic transfer from a central computer server to a subscriber's computer over common carriers or wide area networks is an attractive proposition. Similar considerations apply to the distribution of software program updates, although cost and frequency of issue are not such serious constraints. A problem faced in both situations is that of incorporating the received material with the original material so that a fully integrated publication, information database or software program is obtained by the user.

Another class of electronically distributed information product comprises home shopping catalogues of mail order products distributed on optical or other digital data storage disks which may contain text, sound and images from printed catalogues or uniquely created material, for example software application demos. To applicant's knowledge and belief, available products lack any computer order placement capability, requiring orders to be placed by voice call.

Communication between remote computers, not directly interconnected by umbilical cable or a wired network, is

enabled by a wide range of hardware devices and software drivers, utilities, applications and application modules. Telephone modems that couple a computer with the telephone network are familiar devices. RF modems that couple computers into wireless networks are less familiar but are beginning to appear in consumer devices known broadly as personal information communicators (PIC's) of which personal digital assistants (PDA's) such as Apple Corp.'s NEWTON® product are a first generation. New kinds of digital communications devices can be expected to emerge as digital technology replaces analog transmission.

General-purpose, online, modem-accessed, electronic information services, such as PRODIGY, COMPUSERVE and AMERICA ONLINE (trademarks), and some Internet services, provide wide access to timely information products from a central server, but are limited and complex. They provide no means for the integration of downloaded information with information products offered on disk or CD, and provide only rudimentary facilities for local viewing and search of downloaded files.

Such online information services provide their own user interface which is generally unlike that of a disk or CD-based information product, and can be customized very little, if at all, by a publisher using the service for product distribution.

Online services are oriented to extended online sessions which require complex user interaction to navigate and find desired information objects. Initial setup and use is rendered complex by requirements related to extended session use of data networks and the frequent need to navigate across the network, and through massive data collections, to locate desired data items. General-purpose online information services do not provide a suitable medium for electronic information publishers to distribute updates, and the like, because of limited interface flexibility, because a publisher cannot expect all their customer base to be service subscribers, and because of cost and payment difficulties. Such services are centered on monolithic processes intended for national use by millions of subscribers which processes are not readily adaptable.

Online service charging mechanisms are also inflexible and inappropriate for most individual information products, requiring monthly subscription fees of $5–10 or more, plus time charges for extended use, which are billed directly to users, after a user sign-up and credit acceptance process. Such cost mechanisms are too expensive and too complex for distribution of many products such as magazine and other low cost update products. They do not presently permit a publisher to build an access fee into a purchase price or a product subscription.

Recent press announcements from corporations such as AT&T, Lotus, Microsoft and MCI describe plans for new online services providing what are called "groupware" services to offer rich electronic mail and group collaboration functions, primarily for business organizations. Although offering multiple electronic object transport operations such services are believed to have complex setup procedures and software requirements and complex message routing features and protocols, and to lack interface flexibility. Accordingly, they are not suitable for mass distribution of low cost electronic information update products and cannot achieve the objectives of the invention.

Communications Products

Many software products exist that enable one computer to communicate with another over a remote link such as a telephone cable or the air waves, but none enables a vendor

substantially to automate common carrier mass distribution of an electronic information product to a customer base employing multiple heterogenous systems with indeterminate hardware and software configurations. Two examples of popular such software products are Datastorm Technologies, Inc.'s PROCOMM (trademark) and CENTRAL POINT COMMUTE (trademark) from Central Point Software, Inc. which are commonly used to provide a variety of functions, including file transfers between, interactive sessions from, host-mode services from, and remote computer management of, modem-equipped personal computers wired into the telephone network.

Counterpoint Publishing's Federal Register Publications

Counterpoint Publishing, (Cambridge Mass.) in brochures available to applicant in November 1993 offered electronic information products entitled "Daily Federal Register" and "CD Federal Register". "Daily Federal Register" includes communications software and a high-speed modem. Apparently, the communications software is a standard general purpose communications package with dialing scripts that are customized to the needs of the Federal Register products. Accordingly, the cost of a communications package license which may be as high as about $100 at retail must be included with in the product cost. Also, Counterpoint Publishing avoids the difficulties of supporting various modems by providing its own own standard modem, with the product, building in a cost (about $100–200) which renders this approach quite unsuitable for mass-market distribution of low cost electronic information update products. The resulting product is not seamless either in its appearance or its operation because the communications software is separately invoked and used, and has its own disparate look and feel to the user.

The "CD Federal Register" provides the Federal Register on CD-ROM at weekly intervals for $1,950.00 and CD-ROM disks are shipped to customers as they become available. Back issues are $125 each. Updates are provided by shipping a disk. The Federal Register is a high-value product intended for specialist, business, academic and governmental users. Distribution of updates on CD-ROM, as utilized by Counterpoint Publishing, is not a suitable method for lower value products such as a weekly news magazine, because of the associated costs. Shipping delays are a further drawback.

While the two product "CD Federal Register" and "Daily Federal Register" might be used together, at an additive cost, to provide a combination of archives on CD-ROM plus daily updates obtained and stored until replaced by a new CD-ROM, based on information available to the present inventor it appears that the two products must be used separately. Thus they must apparently be viewed, searched, and managed as two or more separate collections, requiring multiple steps to perform a complete search across both collections, and requiring manual management and purging of the current collection on hard disk by the user.

Xcellenet's "REMOTEWARE"®

Xcellenet Inc. in product brochures copyrighted 1992 and a price list dated Aug. 16, 1993, for a "REMOTEWARE"® product line, offers a range of REMOTEWARE® software-only products providing electronic information distribution to and from remote nodes of a proprietary REMOTEWARE® computer network intended for use within an organized, corporate or institutional data processing or management information system. The system is primarily server directed, rather than user initiated and requires an expensive program (priced at $220.00) to run at the user's node whereas the present invention addresses consumer uses which will support costs of no more than a few dollars per node.

Further, REMOTEWARE® is primarily intended to be used with other REMOTEWARE® products at the node which other products provide a range of user interface and data management functions, at significant additional cost, each with their own separate user interface presenting a standard REMOTEWARE® look and feel. In addition, the nodes require a sophisticated central support and operations function to be provided, which may be difficult for an electronic information publisher to accomplish and add unacceptable expense.

REMOTEWARE® is overly elaborate to serve the simpler objectives of the present invention. Designed for the demanding needs of enterprise-wide data processing communications, the client or node package provides many functions such as background operation, ability to receive calls from the server at any time, ability to work under control of the central server to survey and update system software and files and an ability to support interactive sessions, which abilities are not needed to carry out the simpler information transport operations desired by the present invention. Such capabilities may be desirable in an enterprise MIS environment, but are not appropriate to a consumer or open commercial environment, and bring the drawbacks of complexity, cost, and program size, which may put undesirable operational constraints on the user (and perhaps even compromise the user's privacy). REMOTEWARE® is too costly and complex for mass distribution of updates to periodicals, cannot be shipped invisibly with an electronic information product and requires specialized server software and operations support that would challenge all but the largest and most technically sophisticated publishers. Accordingly, REMOTEWARE® is unsuitable for widespread use as an economical means of distributing updates for a variety of electronic information products.

Although it has wider applications, a significant problem addressed by the invention is the problem of economically distributing updates of electronic information products to a wide customer base that may number tens or hundreds of thousands, and in some cases, millions of consumers. At the date of this invention, such a customer base will normally include an extensive variety of computers, operating systems and communications devices, if the latter are present, all of which may have their own protocols and configuration requirements.

While an electronic information product vendor might consider licensing or purchasing an existing commercial communications product for distribution with their publication product to enable remote, diskless updating, the high cost of such a solution would generally be unacceptable because a communication package includes a broad range of functionalities not required for the vendor's particular purpose, for example, remote keyboarding. Significantly, a commercial communications package is not susceptible to customization of its user interface and may have its own configuration requirements and installation requirements, with regard to directories, device drivers and the like, which are incompatible with other vendor or user requirements or are simply a nuisance to the user. Thus, a commercial communications product in addition to its cost, cannot be satisfactorily integrated with an information product.

There is accordingly a need for computer-implementable information transport software to enable simple, economical and prompt mass distribution of electronic information products.

SUMMARY OF THE INVENTION

This invention solves a problem. It solves the problem of enabling simple, economical and prompt mass distribution of electronic information products.

The invention solves this problem by providing a computer-implemented information transport software module usable with any of multiple electronic information products for mass distribution of electronic information objects to users of a diversity of uncoordinated communications-equipped computer stations. The information transport software module is readily customized to an individual information product to have a user interface in said information product for activation of automated transport of an information object between a remote object source and a user's computer station. The information transport module contains user communications protocols specifying user station functions of the automated object transport and the object source is supplied with source communications protocols specifying source functions of the automated object transport. The source communications protocol is co-operative with the user communications protocol and knows the characteristics of the user communications protocol, so as to be able to effect the information object transport in unattended mode after initiation.

Preferably, for economy and simplicity, the information transport component is supplied for incorporation in an information product as a free-standing embeddable component comprising only such functionality as is required for the aforesaid information object transport operation as that operation is described above and as further elaborated herein. In a preferred embodiment, by limiting available functionality to predetermined transport operations, for example to information object transport between the user's address and one or more pre-specified remote addresses, or to transport of a pre-specified information object or objects, or by making both such limitations, a lean and efficient information transporter product can be provided. This enables an information product vendor to supply an automated, or unattended, update or other information transport facility to a mass market of computer users without the complexity and expense of proprietary network or communications software packages, or of the vendor developing their own transport software.

In a local area network, users communicating across a common medium such as ETHERNET (trademark), or TOKEN RING (trademark) can enjoy the relatively expensive benefits of coordination of traffic between users, and to and from network services, which benefits are provided by a network operating system such as LANTASTIC (trademark, Artisoft Corp.) or NETWARE (trademark, Novell, Inc.). In contrast, a mass market of computer users lacks coordinating means for the facilitation of remote communications between the users and a would-be provider of services to those users. The inventive information transport component, or transporter, efficiently fills that need. While the invention might be implemented for transport across a local area network, such use would probably be incidental to the provision of other services and may not be needed having regard to the sophisticated functions usually provided by relatively much more expensive local area network communication systems, for example, a network file system providing distributed file management functions permitting simple transport of files between network stations.

Typical communications equipment comprises a modem, but other cards and devices enabling remote communication between computers may be used, such as devices or means permitting communication in a digital rather than analog realm, for example, ISDN or ATM interfaces when they become commercially viable.

Preferably, the user communications protocols specify parameters such as a source address, which may be a common carrier address, such as a telephone number, and object parameters such as file name or names, file size, location content and format are specified, as appropriate, in either the user communications protocols or the source communications protocols, or both. Such object specification can be listed in an object manifest stored at the user's station, which preferably, for better control of the transport operation, is sent to the remote object source as a verifier.

By pre-specifying the desired transport functions to both ends of the transport operation, the user and the object source, a simplified, easy-to-use, automated transport operation which conveys an information object in unattended mode, after initiation, can be provided to any user.

The inventive information transport module provides an information product vendor with simplicity, modularity and generality enabling information fetch operations to be easily executed by novice users, and permitting inclusion in a wide range of information products with a minimum of customization. The invention is accordingly most suitable for electronic publishers to employ to enable their customers easily to update information products such, for example, as periodical collections, patent collections or software furnished on optical, magnetic or other storage devices.

In a preferred embodiment of the invention, the information object is pre-identified and integratable with the information product to which the transport module is customized to provide an augmented information product and the information transport component comprises:

a) a fetcher module configured to fetch said pre-identified object from said object source employing a pre-specified common carrier address stored in said fetcher module;

b) a communications manager to establish and manage connection to said object source under control of said fetcher module and with the assistance of said user and source communications protocols; and

c) a fetched object integrator to locate a fetched object in a preset file area accessible to and known to said containing information product;

wherein said object pre-identification, said common carrier address and said preset file area specifications are stored in said software component, whereby a workstation user of said information product can automatically effect transport and integration of a pre-identified object from said object source to create an augmented information product at said workstation.

In this embodiment, any user can, easily and with varying degrees of automaticity, up to complete automation after initiation of transport or upon arrival of a scheduled transport time, obtain an update object and smoothly integrate it with an original product or product shell.

In a highly automated embodiment a containing information product, complete with transporter, is pre-coded with an update, reporting, or other schedule and, referencing the user's system clock, prompts the user for initiation of a transport operation at a scheduled date after distribution of the containing product, or fetches a schedule. If the user's system is shut down when the pre-scheduled date arrives, such prompt may be made at the first system boot or product use after that date.

The invention provides a closed-ended information transport operation between an information object source and any subscribing user, with no special commands or menu selections, which functions efficiently and, within the general parameters of an operating system's required environment, operates independently of the user's system

configuration. Information transport operations are carried out automatically between communications modules that know what to expect from each other, avoiding difficulties arising from open-ended communications with a wide variety of users employing a diversity of heterogenous systems.

In another aspect, the invention provides a method of distributing predetermined electronic information objects from a remote object source to users of a diversity of uncoordinated modem-equipped computer stations, said method comprising:

a) supplying said users with an information transport module containing user communications protocols specifying user station functions of an automated object transport operation; and

b) supplying said remote object source with a source information object and source communications protocols specifying source functions of the automated object transport operation, said source communications protocol being co-operative with the user communications protocol to effect said information object transport operation;

whereby said transport operation can proceed automatically after initiation at said user's station.

The inventive distribution software module and the original information product are linked together to interact seamlessly. It is possible for transport of the update to proceed in a high level format facilitating integration of the update object with the original product, and the invention also provides methods and software for effecting such integration.

A broad objective of the invention which can be fulfilled by the methods and products disclosed herein is to allow a computer user to fetch and use an information product update, or even an original information product for which they have previously received a transporter kit, with a minimum of effort, and preferably with the impression that the fetch function is an integral capability of the information product itself, rather than being executed by a separate or separable component.

Another objective is to enable information transport to be easily effected across any of a selection of media or carriers, desired by the containing information product supplier. To this end the information transport component can provide protocol selection means for selecting media for real time communication between said user and said remote object source employing a selection from a set of open-ended network technologies and network providers, said communication means being selectable without substantive change to said containing information product.

In preferred embodiments, after setup of a containing information product and a simple menu-selection activation of a transport operation to occur immediately or at a subsequent date, or time, and subject to the occurrence of error conditions, the information transport component effects the transport operation in an unattended manner, or without user intervention, through the steps of modem activation, dialing, network transit, handshaking with the object source, file specification, file importation, termination of the call and return of control to the containing product.

Preferably, additional transport-ralated steps such as sending back verification of receipt of the fetched file to the object source, inspection of the fetched object and comparison with a pre-existing manifest for verification of object parameters, and any necessary unpacking and decompression are effected automatically, in an unattended manner without user intervention. For seamless use of the object, it is also preferred that application file specifications, any necessary location or relocation of an object file or files, and any reindexing, index creation or other product integration function that is required to enable the user to utilize the fetched object harmoniously with the original information product, be performed automatically in unattended manner without user intervention, or with minimal user confirmation that one or more steps of the procedure should be executed.

Should errors be detected, if critical, they are reported to the user, and possibly also to the object source. If a detected error is potentially recoverable, the novel information transport component preferably takes action, without seeking user confirmation (although in some embodiments confirmation could be requested), to correct the error, for example by redialling a phone call a specified number of times, or by re-running an object fetch operation. Should a new fetch object still fail to meet manifest specifications, deviations may be reported back to the object source with the user being alerted and, possibly recommended to make a phone call.

Preferably also, the information transport component or "transporter" performs a containerized, standard transport operation, which is transparent to any high-level formatting of the transported information object, and standard in the sense that the transport operation can be essentially repeated for a wide variety of different information objects.

Preferred embodiments of the information transport component can pack or unpack, compress or decompress, and send to or fetch files from specified locations. The transporter allows the containing information product to be set up automatically to effect high-level integration of indexes and navigational structures by letting the containing product have control when needed to import or export (and encrypt or decrypt) objects.

Preferably, the transporter has no direct effect on the content of the data object. Such transparency is advantageous in avoiding interdependency between the transporter and possible use of novel data structures, encryption or copy-control methods, or the like, by the containing product. For example the transporter need not know (and possibly jeopardize) any encryption technique.

In preferred embodiments of the invention, the module is self configuring and has the ability to scan the user's system, and preferably identifies the user's modem, or other system components or configuration software, and automatically set protocols such as the baud rate, bits parity and the like. Relevant auto-configuring capabilities and software that may be employed in practicing the invention are offered or promised by Intel Corporation in a brochure entitled "Intel Technology Briefing: Plug and Play" copyrighted 1994, the disclosure of which is hereby incorporated herein by reference thereto.

Preferably, the novel electronic information transporter is seamlessly embedded in the containing product so that an end user is unaware that the transporter can exist separately from the containing product. However, it is a valuable feature of the invention that the transporter be separable from the containing product to be us able with other containing products.

New or improved electronic information products are made possible by the novel information transporter disclosed herein, for example, CD-ROM-based products updated from online services, updatable periodical magazine collections, catalog-based computer shopping with order entry and optionally, order confirmation.

Recently Contemplated CD-ROM Products Updatable from Online Services

A CD-ROM-based product with online service updatability called "MICROSOFT Complete Baseball"

(MICROSOFT is a trademark) was announced by Microsoft Corporation apparently on Mar. 1, 1994, with a Jun. 15, 1994 availability date. A product brochure received by the present inventor on April 26 describes a multimedia history of baseball which can be updated with daily scores from an online service, by modem. Nothing in the s ales materials suggests any separable information transport components marketable for use with other information products.

In late April 1994, CompuServe® (trademark) online information service announced plans for a CD-ROM information product to be used in conjunction with its online service. The CompuServe® CD-ROM information product online service is usable only with that service, and requires users of its online component to be CompuServe® member/subscribers, on terms such as described above, which terms restrict the CD-ROM product's marketability. The CD-ROM content and user interface is limited to that provided by CompuServe®. Accordingly, such a dedicated CD-ROM service is not a satisfactory solution to independent publishers looking for economical update means, because they will be limited to whatever user interface and data management flexibility the online vendor may provide which will substantially restrict any creative look-and-feel identity the publisher may have provided in their own product. Thus the CD-ROM product is described by CompuServe® in the statement: "It is, essentially, a new window on CompuServe . . . " This product description does not suggest an ability to obtain updated online information for integrated local, offline use with an original information product stored on the CD-ROM, as is provided by the present invention.

In addition to CD-ROM-based products, various new information distribution methods and services are made possible by embodiments of the present invention. The object source can be a remote server equipped with a cooperative communications module closely molded to work effortlessly with the information transporter for distributing objects to a wide base of users. Such a remote server can be linked to a vendor or gatewayed to other information object sources or electronic publishers, and exploit its smooth and efficient information transport capabilities to act as a distribution point for such vendors, sources or publishers.

Thus, the invention further comprises such a special-purpose server designed for use with the novel information transporter and the special-purpose server can be established as a distribution service for publishers who incorporate the information transporter in their products. The invention also provides a method of operating a server to provide such a software service and server-enabling software.

BRIEF DESCRIPTION OF THE DRAWINGS

One way of carrying out the invention is described in detail below with reference to drawings which illustrate only one specific embodiment of the invention and in which:

FIG. **1** is a schematic diagram of one embodiment of an information transport software component according to the invention installed in a computer workstation and communicating with a complementary centrally located server-resident software module for mass distribution of digitized electronic information objects;

FIG. **2** is a flow block diagram of an information transport operation performed by the software component and module of the embodiment of FIG. **1**;

FIG. **3** is a schematic diagram of a server-based electronic distribution service employing an inventive information transport software component;

FIG. **4** is a further schematic diagram of the service illustrated in FIG. **3**;

FIG. **5** is a schematic diagram of a prior art communications product employed to transport an information object between a user and a remote server; and

FIG. **6** is a schematic diagram s similar to FIG. **5** showing, in a comparative manner, some of the benefits that can flow to a user when an information transport software component, such as that described with reference to FIG. **1**, is used for a similar transport operation.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring to FIG. **1**, the inventive software component is schematically shown in operative mode installed at a user's computer workstation. The workstation is communications-equipped for communication with remote services, for example by modem, which services are also shown schematically. Only relevant software and hardware components of the system are shown.

Relevant components at the workstation comprise operating system services **10**, a containing information product **12**, an information transport component or module **14**, herein also referenced as a "transporter" which may be a stand-alone product or, in preferred embodiments is embedded or contained in the containing information product **12**. Information transport component **14** provides a general purpose facility for sending and fetching information objects between an end user's computer (the client) and a central server. Information transport component **14** is not customized to the containing information product **12**, but is intended to be used in conjunction with any of a wide range of electronic information products.

Operating system services **10** provide capabilities for the containing information product **12** and the information transport component **14** to access a readable information storage device **16** which may, for example, be an optical disk drive such as a read-only CD-ROM where product information **17** is stored. In addition, a read/write information storage device **18**, for example, a conventional hard disk is accessed via the operating system services **10** for storage of a fetched additional information object **26**. The storage media used for hard disks and the like are often described as nonvolatile and the type of storage is frequently referenced as "permanent". However, the more recently used term "persistent storage" which references the manner of storage as well as the physical storage media and distinguishes from transient storage where objects may be automatically erased after some interval, or event, without supervisory control or awareness of the erasure, is a more suitable descriptive term for the purposes of the present invention.

As necessary, different, or modified, information transporter components **14** can be supplied for users of different operating systems or system families, notably DOS (available in several versions, for example from Microsoft Corp, IBM Corporation, Novell, Inc.) Windows (trademark, Microsoft Corp.), Apple Computer Corp.'s operating systems, possibly IBM Corporation's OS/2 (trademark), and any distinct operating systems developed for personal digital assistants, pen-based computers and the like.

Information transport component **14** also uses operating system services **10** for external communication with a communications network **20** through which the information transport component **14** can access a remote server **22**, or server-client network, supporting a data storage device **24** where desired additional information object **26** is located.

Communications network **20** can be any electronic distribution system suitable for transporting information objects **26** including wired and wireless common carriers such as telephone networks, cable television systems or networks and mobile telecommunications or data communications networks and extends also to emerging and future systems of providing electronic communication between users of diversified equipment. The term "common carrier" is used herein to embrace all such data communication systems as will reasonably meet the purposes of the invention. The term "modem" is used herein to embrace any network interface device enabling a user station to communicate on such a communications network **20**.

While the containing information product **12** can take many different forms, as described herein, and as will also be apparent to those skilled in the art, a preferred embodiment is that of a periodically issuing publication or publications, for example, a news magazine or a collection of patents. Again, the additional information object **26** could be any information of interest to the user, having some relevance to the containing information product **12**, but the invention and its unique capabilities enable the additional information object **24** to be fully integrated with the containing product **12** in a manner that can be automated to be transparent to the user.

The inventive information transport component **14** is designed to require a minimum of user input. A bare minimum will be a user's ID which can be entered by the user in a product setup and automatically accessed for information transport, or could by pre-loaded by the vendor from data supplied by the user at purchase.

A product ID is preferably pre-loaded into the containing information product **12** by the information product vendor or publisher to be available for use by the information transport component **14**. However, even this may not be required. In an alternative embodiment, the product ID can be automatically incorporated into the product in a product replication process that permits individualized coding of unique ID's. In most cases, a user-actuated menu selection is provided in the containing information product **12** after integration with the inventive information transport component **14** to activate transport of an additional information object, and preferably, selection of transport activation drops down a menu of transport choices such as "FETCH UPDATE", "FETCH CATALOG OF UPDATES", "SEND DATA" and the like, each of which then runs automatically upon selection.

Updating can also be totally automatic, and other than an obviously desirable user notification, be completely invisible to or transparent to the user, running in background on their system, while the user's screen is available for other processing such as running the containing information product **12**. Where updates are made available on a known schedule, a totally automated product can be provided that fetches an update without any user intervention, on the specified release date, or as soon thereafter as the user's system, or the containing information product **12**, is activated. In practice, most users will probably prefer an opportunity to confirm that the fetch transaction should proceed. A preferred embodiment monitors the user's system clock and alerts a user to the arrival of an update release date and asks the user to confirm that the system should seek and fetch the scheduled update, if available.

Thus, the invention is particularly suitable for importing updates of information or information processing products, such as periodically issuing literature, or software upgrades. Accordingly, additional information object **24** preferably comprises updates which can be integrated with the information product **12** to provide, for example, a coherent body or continuous sequence of materials that can be commonly searched and indexed preferably in a manner giving the user the appearance of a common logical file formed from physically distinct files. The appearance of integration can be achieved by searching new and then old indexes in series and making the search and navigation logic of the containing product smart enough to combine new and old information. For example a new object can have an index file similar to that for the original information product **12**. A search engine can first search the new index, then the old one, and then produce a combined set of results. Preferably, the files are not actually merged or otherwise combined as to do so could be unduly complex.

As shown in FIG. **1**, the containing information product **12** comprises a user interface **28** enabling the user to view, search, excerpt and print or otherwise export or process selected information items from product information **17**. The user interface **28** provides standard information product features, as conventionally supplied by the product publisher, supplemented by appropriate fetch or send options to activate the features of the inventive information transport component **14**.

Also shown in FIG. **1** are a database management module **30** and a data structure definition module **32**. Database management module **30** provides retrieval-oriented database processing of the information product including indexed searching and selective retrieval capabilities using one or more index keys such as an issue or item number, or full text searching, and may provide hypertext and hypermedia linkages. The data structure definition **32** provides the database structure of relevant files as classified by field or element, name, type, size and the like. After successful completion of a fetch operation, control is returned to containing information product **12** to process the new information in essentially the same manner as the original information, or in any other manner for which it has been equipped.

Major modules comprised in the inventive information transport component **14** are a user interface **34**, a communications module **36** and fetch-send protocol **38**. In addition, the information transport component **14** preferably comprises its own built-in application programming interfaces (APIs) such as a user interface API **40** and a communications API **42**, enabling the information transport component **14**'s user interface and communications modules respectively, readily to be incorporated with, or plugged into a wide range of containing information products **14**. Such incorporation, in the currently best known embodiment of the invention, is effected by software engineers familiar with and having access to the containing information product **12**, but future developments may enable the incorporation process to be effected by skilled users.

References herein to an applications programming interface (API) will be understood to embrace any program interconnection technique which supports direct, seamless interaction between one program and another, including procedural calls, object encapsulation, or emerging techniques like Microsoft Corp.'s Object Linking and Embedding (OLE) or Apple Computer's Open Doc.

API **40** is responsible for providing means for the user to interact with the information transport functions of the invention and interface as seen by the user and API **42** is responsible for handling internal processes of communications and data management.

The APIs **40** and **42** are intended to enable the information transport component **14** to be used by a range of product

programs controlling a variety of information products and to enable each API **40** and **42** to be free to exercise flexibility and creativity in extending its associated user interface **28**, data management module **30** and database structure **32** to fully address the provision of transport functions for the purposes described herein.

API **42** operates on a transport function level involving high level interactions between the containing product **12** (or the optional user interface) and the transporter **14** before and after communications while the detailed low-level interactions between the transporter client and the server during communications are handled by fetch-send protocol **38**, without involvement of the containing product **12** or the user. "High level" is used to refer to a level at which software interacts with a user, typically in simple, readily comprehensible, function-oriented, graphic or everyday language terms, while "low-level" refers to a level of detailed procedural interaction with an operating system, or device (modem, port etc.) in obscure program or machine language terms incomprehensible to most users.

Fetch-send protocol **38** is, in the preferred embodiment shown, a component of a novel client-server communications procedure designed to manage the transaction-oriented transmissions required to achieve satisfactory transport of desired server stored information objects, and optionally, central reporting of user information in a predetermined format. Alternatively, one or more existing protocols could be used.

Preferably, the API's **40** and **42** and the fetch-send protocol **38** are structured to use a manifest list to control the exchange of information objects. The manifest list can be provided in fetch-send protocol **38**, and can be forwarded to remote server **22** to provide better efficiency, error control, and management of the operation. Alternatively the manifest list may remain resident at the user's station. The manifest is valuable operating at the client station, at the API level, to specify the actions required during a transport session and can in one embodiment comprise a list of send and fetch operations which are individually controlled.

This software mechanism, employing novel communications procedures and applications interfaces that reference an object manifest, provides a new way for performing a wide variety of information exchange functions in a simple, standardized and economical manner.

API Functions: 1) Product Setup

In preferred embodiments, API **40** and API **42** include a product setup routine of an application-specific configuration, which is used by the publisher or product developer, prior to publication, to establish seamless compatibility between the containing information product **12** and the information transport component **14** for smooth execution of desired transport functions. A completion status code is also specified.

The application-specific configuration posts user and product ID information, as needed to process password or other access code authentication and posts files information, including designation of an application work directory and a transporter work directory for performing the transporter functions of information transport component **14**.

Additionally, the application-specific configuration sets up an appropriate decompression (or compression for send objects) technique according to the expected format and condition of fetched information objects **46**, which information is pre-coded into communications component **36**.

The application-specific configuration established through API **40** selects either a standard user interface, as furnished with information transport component **14**, or an application-controlled user interface. Control settings are established for connection problem handling, disk error handling, abort and server condition handling, access denial, unavailability of information object files and any other error situations which may occur during transport.

If desired, optional, advanced controls for scheduled automatic calling can be included in the application-specific configuration used in preparing the containing information product **12** for publication.

Preparation of containing information product **12** and incorporation of information transport component **14** therein, with an application specific configuration, as described is carried out prior to publication to build a customized, ready to run version of the product with automated update capability.

Communications API **42** establishes a product-specific transport method choice list for selection of an appropriate file transfer protocol as between direct dial, data network dial, and other modes of transport. Communications protocols specify necessary connection parameters such as access number and network addressing or other routing information. Optional script choices can provide for different modes of transport.

These product-specific configurations and protocols enable information transport component **14** to be packaged in executable form with containing information product **12**, with all necessary product-specific components and settings, including a standard user interface if selected, ready for inclusion in the product package.

If desired, at the option of the information product publisher, a standard user interface may be included. Such an optional standard user interface can have all facilities needed to select transportable objects from a predefined list, perform all user setup functions, and invoke information object transport.

Additional options are standard software that would allow the user to search, view and print the transported objects totally independently of the user interface and database search components of the containing product. Both such options enable a publisher to exploit the inventive transport product for efficiently and economically providing updates without having to make changes to the publisher's containing product, simply by configuring the transporter or information transport component **14** and physically including it, and the optional components, within the containing product.

A standard viewer might handle only ASCII text, but it preferably could provide for other useful formats such as standard word processor, spreadsheet or database formats, or multimedia formats such as video, sound and HTML (hypertext markup language), a format becoming popular on the Internet and which provides the ubiquitous Web pages for the Internet's World Wide Web.

API Functions: 2) User Setup

Compatibility with the user's system is effected by API **40** establishing a user-specific configuration, and creating or updating the necessary control files.

Parameters established in the user-specific configuration include a setup ID number to permit use of multiple setups, for example, for different transport options, and a product ID number.

The user-specific configuration posts user ID information and a password or other access code authentication and posts files information, including disk and drive designation for work and data directories. Autocall options and a completion status code are also specified.

API **40** provides information for communications module **36**, specifying a user communications protocol for the user's

hardware, operating system, line configuration, and so on. Thus, for a standard telephone connection, comport, speed (baud rate), interrupt settings, modem type and control strings, dial prefix, dial 9, pulse or tone, call waiting shut-off, and the like are specified, as appropriate. Additionally, the user communications protocol includes access number and connection parameters, optionally with script selection for routing choices via data networks, and so on.

The resultant user-specific configuration and communications protocols generated through API **40** create a setup ready to call and places it in the designated transporter work area.

A validation procedure checks entries and reports obvious errors in parameter settings.

Preferably, multiple product ID setups are provided to enable multiple information products to use the transporter with an appropriate, compatible transporter version. Preferably also, the user-specific configuration accommodates shared use of the transporter work areas by multiple information product applications resident on the same user's system.

Mechanism of Fetch-send Protocols **38** (user) and **44** (server)

User fetch-send protocol **38** working in cooperation with server fetch-send protocol **44** controls the desired information object transport function, calling remote server **22** and exchanging data objects. It performs or supervises communications between the user's system and remote server **22**.

Communications module **36** uses a setup ID number specified through API **40** or **42**, selects which setup to use for a call, calls remote server **22** using protocol **38**, and in a preferred embodiment, sends an object manifest comprising a send object list, a fetch object list or both. Such manifest is created under control of user interface **28** from a pre-existing set of choices supplied with the product or obtained during previous update operations, or both.

Alternatively fetch-send protocol **38** may refer to a pre-existing manifest list stored at the user's station, or may be directed by remote server **22** to select one of multiple pre-existing manifest lists stored at the user's station. As another alternative, although it is convenient and advantageous to transmit the manifest list to the server **22**, the relevant status and management information can simply be used locally by communications module **36** and be integrated into the individual fetch and send protocols.

A send object list comprises object action codes specifying the type of server action required, if any, object names, object sizes and response object size, if any. A fetch object list comprises object names, object sizes and an object availability date.

A completed object manifest is employed to convey the status of the transport operation and to provide for additional information transport, if desired. The completed object manifest adds the following to the request object manifest: send object additional information; object acceptance codes returned by server **22**; time of acceptance; and a response object name, if called for by the object action code.

For a fetch operation, the completed object manifest adds the following to the request object manifest: fetch object additional information; a fetch confirmation or failure code; the time of completion or failure and a revised availability date if the requested fetch object was unavailable.

If a scheduled update or polling option is present and selected, a scheduling or polling indicator is included, and a completion of processing or import function to call through API **42** is specified.

A completion status code terminates the fetch or send operation and returns control to the information product application or the provided user interface.

Information Transport using Communications Module **36**

Communications module **36** employing the described fetch-send mechanism comprised by cooperating protocols **38** and **44** performs the functions necessary to complete an information transport operation, as described herein, under a variety of circumstances, with tolerance for a common range of error conditions, open drives, inadequate disk space, lost line connections and the like, without losing control of the user's system. Using correct, verified ID, naming and routing information, the information transport operation employing the inventive information transport component **14** is less error-prone than many computer users would be were they effecting the transport operation with conventional technology requiring them to enter routing and storage information and the like, manually.

Communications module **36** verifies that all send objects are as specified, that all fetch objects are scheduled to be available, verifies that sufficient disk space is available for all fetch objects and for compressed transmission copies of all objects, and returns an error report if any of these requirements is not fulfilled.

Communications module **36** performs communications, then returns a completed object manifest, and logs all activity in a transporter log file. If an optional scheduling/polling feature is selected, the communication is deferred until the scheduled time.

These general objectives are achieved by carrying out the following process steps after an application (or optionally a transporter user interface) requests a transport function:

1) Local validation of the request returning a failure code if the request is improperly specified.

2) Compression of all send objects for transmission and placing them in the designated transporter work area.

3) Connection attempts to remote server **22**, returning a failure code if necessary. Connections are made via phone line or network. The system handshakes and identifies the call to the server.

4) Presentation of the object manifest, if utilized, for validation and action.

5) on receiving a go-ahead, transport of each send object, logging each as sent, and receipt of object acceptance codes from the server and logs them, when received.

6) Receipt of all fetch objects from the server, placing them in the transporter work area, and logs them as received. Fetch object names may be precise, or generic or alias names may be used to request a latest installment.

7) Receipt and logging of a completed object manifest from the server. (If receipt of response objects is implied by the action codes, first receives a revised object manifest, and fetches the response objects, then receives the completed object manifest.)

8) Disconnection from server.

9) Decompression and unpacking of all fetch objects into application work area, and logs completion status.

10) Returns control to the application (or optional transporter user interface).

The product checks the completion code, and completed object manifest to deal with any error conditions. The application performs any required import processing on fetched objects to integrate the data and indexes with prior data, as desired, to enable seamless use. If desired, import processing can include, or offer as a user selection, file maintenance functions relevant to the information product including, for example, file purging to remove obsolete

information files and preserve the user's storage space. Specifications of files to be deleted can be included with the original product or with a fetch object. In either event the responsibility for accurate specification is passed to the vendor, relieving the user of the risk of making erroneous deletions and anxiety attendant thereon. After such import processing the containing information product (or the optional separate user interface) then returns control to the user for use of the received data.

Those skilled in the art will appreciate that the identification of files in the object manifest, or for file maintenance functions at the user station, or for any other purpose of the invention, can be effected generically, for example by using wild card characters, as is customary in file specification, and which effectively permits multiple objects to be specified as a class related by file name characteristics, or related individually, thereby providing options for specifying transport of such class of multiple objects to proceed at one time or in a series of transports over time. Other algebraical identification methods can be used which may reference object versions in series or comparable characteristics.

The foregoing steps are illustrated in the flow block diagram of FIG. **2**. When containing information product **12** issues an information transport call **50**, setup filter **52** runs setup routine **54** if this is a first call and no information transport setup was run on installation of containing information product **12**. At block **56**, an object manifest is retrieved for pre-transport preparation at block **58**. After prepping, a call to server **22** is established at block **60** and when the connection is made, and a handshake performed, one or more objects is transported at block **62**.

After completion of transport and receipt of a completion manifest, server **22** is disconnected at block **64**, received objects are decompressed and unpacked at block **66** and stored in a designated disk storage location at block **68**. Object storage triggers containing information product **12**'s import processing to assimilate the information update with the original information product at block **70**, following which a completion report is issued at **72** and control is returned to the containing information product **12** at **74**.

Optional Schedule Function

An optional transport function module for scheduled or poll-responsive information object transport can be provided to defer the fetching of an update or to defer another information transport operation to a specified later time, or until called by the server.

The optional transport function schedules a request, waits, then automatically performs the transport operation at the scheduled time. In polling mode, it activates (and, if necessary, interrupts and then reactivates) the user station's ability to receive calls.

Mechanics of the optional transport function include a request for an ID number, an indicator for calling or polling mode and a schedule iterating a call time, a retry protocol, call activation and timing, along with an authentication procedure for the server and a completion status code.

Client-Server Communications Protocol

Communications between the information transport component **14**, functioning as a client, and the server **22** follow a predefined communications procedure having cooperative user components comprising user fetch-send protocol **38** and server fetch-send protocol **44**.

Server-client intercommunication can be broken down into five steps, a) login, b) manifest transmission, c) send operation, d) fetch operation and e) logout, as described in more detail below.

a) Login

Login establishes a session with an authorized client. A handshake process between user protocol **38** and server protocol **44** identifies the user's transporter client system to remote server **22** by product ID and user ID, and a password or other authentication code. A failure reason code is given to rejected clients.

b) Manifest Transmission

Preferably, via user protocol **38**, the user system issues an information object transport request manifest to server **22**. Server **22** verifies its ability to meet the request by returning a manifest acknowledgment specifying which elements will be processed and provides reason codes for declined elements. Alternatively, as stated previously, manifest functions can be listed in individual send and fetch protocols.

c) Send Operation

If the user system outputs a send object, through information transport component **14** and protocol **38**, server **22** receives and accepts the send objects and stores them, identified by product ID and user ID. Error control and retry mechanisms are employed and successful receipt of the send object is acknowledged and logged.

If the action code calls for a response object, the server obtains necessary processing from a pre-designated external source (corresponding to the product ID and action code) and returns the response as a fetch object, called a response object.

d) Fetch Operation

The server obtains requested fetch objects by product ID and object name and forwards them to the transporter at the user. Error control and retry mechanisms are employed and successful transmissions are acknowledged and logged.

e) Logout

The server transmits the completed object manifest to the transporter, confirms and logs receipt, and ends the session.

The Inventive Transporter Compared with a Conventional Communications Product

FIGS. **5** and **6** illustrate schematically the simplicity and ease- of-use benefits the invention provides FIG. **6** to a user **100** in fetching an information object from a remote server **22** as compared with the use of a conventional communications product (FIG. **5**), such, for example, as CENTRAL POINT COMMUTE (trademark) or PROCOM (trademark).

In the prior art embodiment of FIG. **5**, many operations require active participation by the user who, for example, must at least initiate any pre-transport preparation **104** of the information object, such as checking the specifications, checking work space available to store a fetched object and conducting any other preliminary checks. The user has to activate a communications product **102**, specify a call route, and after the call connection is established, specify the objects and initiate a transport operation. Communications product **102**, operating in a cooperative manner with remote server **22**, will execute establish call connection **60** after the call route (phone number) has been specified and will execute transport objects **62** after the objects to be transported are specified by the user. Disconnection **64** is usually effected by a user executing a call termination command, which if the user is inattentive, or inefficient, may be delayed longer than necessary to complete the transport operation, running up unnecessary line or air time charges.

After completion of the transport operations, user **100** has to deactivate the communications product **102** and then initiate any required storing and processing of the fetched product **106**. While some of these steps may be automated via one or more batch files, scripts or macros, a vendor of a containing information product **12** has great difficulty in

furnishing such a batch file or macro for a mass market distribution because of the different systems and communications products encountered in a mass market, which systems and products have a variety of different specifications, performance characteristics and unique, incompatible scripting languages.

Equally, while some more skilled users **100** might be able to write their own batch files without undue difficulty to automate some of these steps. Many users will lack the ability or the inclination to do so. Also the effort would not be justified for a single transport operation. Nor is the result of such efforts likely to match the ease and simplicity of the results achieved by the present invention which enables even a first update to be obtained effortlessly with the software running in unattended mode, after initiation.

FIG. **6** clearly shows how the inventive information transport component **14** relieves user **100** of many tedious communication functions such as activating a communications product, specifying a call route, specifying the objects to be transported and deactivating the communications product. In addition, preferred embodiments of the invention also relieve the user of optional pre-transport preparation **104** and execution of store-and-process-fetched-product **106** if these functions are appropriate to the containing information product.

Referring to FIG. **6**, user **100** selects a transport operation from a user interface screen in containing information product **12**, whereupon the latter calls information transport component **14** to activate transport. Information transport component **14** implements any necessary pre-transport preparation **104** and then, employing its own communications module **36**, and server fetch-send protocol **44**, proceeds in unattended mode, without requiring user intervention to establish call connection **60**, to execute transport object **62** and automatically perform a disconnect **64**, as described herein.

Automatic transport control and disconnection is a useful feature of the invention providing economy of line or air time charges and reducing congestion on the communications carrier. Using conventional communications products, (especially with online services) the duration of the connection may be unnecessarily extended by the delays and potential errors inherent in user control, resulting in increased communications costs and failures. The inventive transporter **14** provides software control of the connection duration, enabling it to be confined to a period sufficient to effect said unattended object transfer, enhancing efficient use of the communications medium.

Also as described, the operation can be monitored or controlled by employing an object manifest and is facilitated by the use of pre-specified addresses and transport characteristics. After satisfactorily completing the transport, the information transport component **14** automatically deactivates and returns control to containing information product **12**, preferably with a satisfactory completion report which containing information product **12** notifies to user **100** through the containing information product **125** user interface.

If the transport object **62** was a product update, optionally a store-and-process-of-fetched-object **106** is initiated by information transport component **14** and execution of the store and process operation may be passed to the containing information product **12**. The user can now use the updated product.

As FIG. **6** shows, when read, in comparison with FIG. **5**, the invention enables a user **100** to be relieved of all duties save for minimal selection and notification functions, while no complex added functionality is demanded of containing information product **12**. Optional store-and-process-or-fetched-object **106** is contemplated as requiring only minimal modification of existing containing information product **12** functions while other more complex procedural and detailed transport related functions are handled by the information transport component **14**.

Some non-limiting examples illustrative of practical commercial and industrial applications of the invention will now be described.

EXAMPLE 1

A News Magazine Distributed on CD-ROM

Some weekly news magazines offer subscriptions to a quarterly CD-ROM which contains multimedia material plus a searchable full- text database of the most recent quarter's weekly magazine issues and enabling application software. Newer issues are not provided until the next quarterly disc is mailed. Accordingly the CDROM electronic magazine product steadily becomes out of date and its value lessens.

The invention incorporates an information transport component **14** with a news magazine product stored on a CD-ROM **16**, to enable a user to fetch an information object **46** in the form of new issues (and their associated search indexes) from a remote server **22**, as they become available, for example weekly. The fetched updates are stored on a consumer's computer hard disk storage device **24**. Because of the size of rich content multimedia files, the updates are limited to text material including full texts of interim issues and associated files such as indexes. Because it knows the storage location of the updates, the next CD-ROM issue can include, as an install option, or upon first access, a request to delete the old now-outdated updates from hard disk **24**, creating space for new updates.

User interface **28** in conjunction with user interface **34** contains code providing a menu selection enabling a user to activate the update fetch operation and then to provide integrated or seamless access to the combined data, searching both the hard disk storage device **24** and the CD, using both sets of indexes, so that the contents are viewable as a single collection, although an additional independent searching/viewing function for the updates could be provided, if desired.

A product setup routine adapts the information transport component **14** to work with the news magazine CD-ROM's existing software for creation of a user interface, searching and viewing. Communications options may be limited to direct telephone dial only. A simple user interface addition controls a setup process allowing the user to enter a unique user ID, provided with each copy of the CD-ROM distribution disk, and to create predetermined work areas on the user's hard disk.

A schedule of updates with names, dates, and files sizes is provided in the containing news magazine product on the CD-ROM and is accessed via user interface **28** in conjunction with user interface **34** to create a fetch object manifest **48**.

Optionally, user interface **28** in conjunction with user interface **34** creates a send object manifest **48** to control transport of user demographics for market analysis or for renewals, or the like, in the opposite direction from the user to the server, with the send operation being triggered whenever the next transport operation is activated, or optionally, by allowing the user to trigger it.

A fetched information object **46**, such as an update, is automatically decompressed and stored on hard disk storage device **18** as additional information object **26** for integration with the original CD-ROM product so that the user can view both the update and the original issues, and run searches across the entire collection.

Optionally, initial location of additional information object **26** may be an application work area location on storage device **18**, and communications component **36** may be pre-set to pass control via API **42** to database management module **30** which will do further processing to integrate additional objects in accordance with the existing database structure **32** to provide a more complete level of integration permitting, for example, viewing of combined menus, nullification of obsoleted items, and cross-linking of hypertext elements.

If a send object has been prepared and included in the object manifest, such as a send object containing user information entered during the install process, or subscription request information obtained from the user, it is sent to server **22** to be stored and identified by product and user ID for appropriate action in due course. Acknowledgement of receipt of the send object is noted by communications component **36** and passed back to the user if such provision is made in user interface **28**.

Both the fetch and send operations are closed ended in the sense of being operations that are pre-described in the original information product and once triggered, can be completed without human intervention of any kind.

To service the automated update facility running at the user's workstation, remote server **22** is set up to accept calls from valid user ID's, and is loaded with new issue text and index files, in the form of update information object **46**, according to a publication schedule.

EXAMPLE 2

Open-ended Fetch of a Supplementary News Magazine Object

Open-ended access to supplemental information objects not described in the original information product can be obtained by providing in the original product means to fetch a directory of added features. This can be used, for example, by a news magazine publisher to provide special news features on an unplanned basis, or each weekly issue could be packaged with a directory of additional features available. The user first specifies a fetch of the new directory, or receives it along with a fetched update they have specified from a user interface menu, and then views the fetched additional features directory and initiates a fetch of a selected additional item or items in a second information object transport operation, using an information object manifest built from the new features directory.

The original, containing product news magazine CD-ROM user interface **28** preferably has provision for importing and viewing any information objects listed on a completed fetch manifest and delivered by the information transport component **14** into the designated work areas. Alternatively, a standard information transport component **14** user interface **34** can be used to provide this function in a less integrated form.

EXAMPLE 3

Retail Catalog on CD-ROM with Merchandise Order Entry at the Server

Multimedia product catalogs with 800 ordering numbers are now available on CD-ROM and also with pre-installed software packages on new computer hard disks. In this example, the multimedia (or text and graphic) product catalog is a read-only information product **17** which can be furnished with an information transport component **14** according to the invention, to facilitate order placement from such electronic product catalogs providing an easier order placing process than has heretofore been possible. Employing the inventive information transport component **14**, a catalog vendor can enable a customer to place the order directly, via modem, without requiring a voice call and ensuing verbal product identification, by pointing and clicking a "Place Order" or "Mark for Order" button on the user's computer screen. The order is transported to remote server **22** using the novel information transport component **14**. Preferably a verification routine is included, requiring order confirmation with a user-supplied password, and possibly keying of the total amount to prevent unauthorized or inadvertent product ordering, for example by children.

Order fulfillment is effected by processing of the information in due course after receipt by the remote server **22** and any additional information required centrally is collected during product setup and held locally for transmission with an order. For example, setup can capture the user's charge card information, shipping address, and the like and create a header for an electronic order form.

When the user clicks the "Mark Order" button, procedures supplied with the user interface **28**, as modified through user interface API **40**, add order item identification information to an electronic order form. When the user clicks the "Place Order" button, user interface **28** triggers a transport request to server **22**, to include the order form as a send information object **46**. Transport of the send object, including the order form, from the user's station to the server is executed employing an object manifest **48**, as described herein.

If not located at a vendor's or merchant's premises, server **22** can forward received electronic orders to the merchant for fulfillment, at appropriate intervals, via a vendor link **50**.

This simple, low cost mechanism for automated order placement, can complement telephone ordering but lacks the credit-checking and inventory status capabilities that are frequently provided by phone. However, such a catalog application could allow the user to request the fetching of an inventory and price update object for use prior to the preparation of an order.

EXAMPLE 4

Merchandise Order Processing and Confirmation Retail Catalog on CD-ROM

A powerful electronic merchandising tool can be provided by providing the user with a full-function order generating capability and employing transporter **14** to transmit a user—created merchandise order, effortlessly and seamlessly, to a remote order-processing server. To this end, server **22** should be interfaced to the necessary merchant processing services for checking and reporting credit and inventory status.

An additional valuable option enables the system to apply pre-specified user instructions, previously obtained through user interface **28**, to determine whether out-of-stock items are to be dropped, back-ordered, or substituted in color or other aspect. This information can be added to the electronic order form object, listed in object manifest **48** and become the subject of a further transport dialog between the user's station and server **22**. In this manner a sophisticated purchase transaction is completed in a substantially unattended manner (save for deciding about back orders off-line), in as

much as the customer does not have to maintain a phone conversation, while fully achieving the capabilities of telephone order placement. A further user benefit can be obtained by the providing a permanent record of the transaction (a stored electronic file) without user intervention. This not possible with telephone ordering.

This novel, automated, modem driven, order placement system effectively shields a merchant from having to deal with the problems of establishing communications with a mass of unknown end user computer systems, while automating the process and relieving the merchant of the costs of telephone sales staff. This aspect of the invention is valuable in avoiding troublesome, support intensive, communications which are subject to rapid technical change as new products are absorbed into the marketplace. In contrast, the merchant's special purpose vendor link **50** to the server **22**, can remain relatively stable, while the customer interface at server **22**, depending upon the sophistication and universality of the API's **40** and **42**, and also upon any emergent communications standards, can be adapted to accommodate a range of future products.

EXAMPLE 5

Further Applications of the Invention Locked Information Products

As discussed in the "BACKGROUND OF THE INVENTION" hereinabove, some vendors, for example Microsoft Corporation, distribute information products in locked, inaccessible form, accompanied by (user-accessible) promotional information and demo versions. The prospective purchaser then calls an 800 number to order the product and is given a code which is entered to unlock the item for use. The inventive information transport component **14** and cooperative server component **22**, can be used to simplify this process, and eliminate the voice call.

The information transport component **14** is used to place the order and as a subsequent step concomitant with satisfaction of the merchants purchase requirements (payment, etc) can, employing a suitable line entry or entries in the object manifest **48**, fetch the access code, as an information object **46**, in the same way as an order acknowledgment or other information update. The user interface and data management components of the distribution CD, or original information product, can be programmed automatically to use the code to unlock the product.

Employing the novel, digital, modem-enabled communications products of the invention, more sophisticated access codes than are suitable for verbalizing to a caller, can be used, and may include small programs or decompression utilities (although these would better be stored in the locked product), or customer-specific coding employing user-derived information. Thus, as a safeguard against fraud, being equipped with specific user or user product information, the access code can be a key or product uniquely matched to the user's locked product copy.

Computer Software Updates: For distribution of updates to software products, the original distribution version of the software product can provide registered users with an appropriate ID code and update schedule. Should the revision be delayed, a revised schedule can be fetched.

Tax or other governmental filings and exchanges: An example of the generality of the inventive information transport system for sending and fetching well-defined information objects of many kinds is in the filing of tax returns. A send information object can be created and manifested to submit electronic tax filings to the IRS, as described above, for electronic product order forms. A fetch object can be created to obtain updated tax forms and the program logic relating to them, and to get information on new regulations. Analogous uses will be apparent to those skilled in the relevant arts of, for example, financial planning and portfolio management systems, to obtain current statistics, place orders, and the like.

Packaging of Transporter with User Interface/Database Search Software Facilities

In a modified embodiment, the inventive information transport component **14** is integrated with a general purpose user interface/database search (UI/DB) software package and tools. Such packages and tools, sometimes referred to as "authoring packages", are now used to produce CD-ROM's and similar information products. Thus a single UI/DB product may contain the inventive information transport component **14**, and be supplied to publishers to be used to develop a family or diversity of information products, as a standard tool box.

A combination of the inventive information transporter product with such UI/DB products could facilitate development of applications by allowing much of the work of integrating a containing product's user interface **28** and database functions **30** and **32** (which could be controlled through the UI/DB product) with the inventive information transport component **14** to be performed once, in advance, by a UI/DB software vendor's skilled specialists, for use in a diverse range of products using that vendor's software. Such integrated offering would be advantageous to both the software vendor (by enriching its offering) and to the software vendor's publisher-customers by facilitating the desired function.

Electronic Product Distribution Service

In a valuable application of the novel electronic information transport products of the invention, remote server **22** can be operated to provide an electronic data product distribution service for multiple containing information products **12**, each equipped with an information transport component **14**, the whole facility providing a complete network distribution service, including network, technical and end-user support. Provision of such a distribution service is greatly facilitated by the novel transporter **14**, described herein, the use of which for each vended product greatly simplifies the problems of handling updates to multiple products. However, such a novel service could also be operated with conventional software communications products by relying upon users of each to execute an appropriate sequence of menu selection and command line instructions to obtain an update by modem via their own pre-existing communications software. Similarly, While special advantages of seamless user adoption and integration into an original product accrue from the use of the inventive transporter to distribute product updates, such a distribution service can be used with advantage to distribute any type of electronic information product.

For many publishers (and for providers of UI/DB authoring software) the task of operating a publicly available server **22**, and of supplying associated technical support to a wide base of customers using a diversity of communications products, even with the simplification benefits provided by the inventive transport product, is a task requiring specialized skills and staffing that a publisher, even one experienced in electronic publishing, will generally lack. Such a specialist capability is intimidating to provide and difficult to cost-justify for the limited number of information products that one publisher can supply.

By providing a new turnkey service or service bureau a specializing, skilled vend or would enable the publisher to a void such burden. A provider of such a novel service can spread the costs of such operational activities and skilled staff across a large number of publishers a nd information products, achieving economies of scale and specialization.

The inventive information transport products extend to software implemented at server **22**, or at one or more clients or satellite servers, of a network served by server **22**, to provide the server-location functions of such an electronic product distribution service. such distribution software can be separately marketed to publishers or UI/DB vendors who wish to operate such a service.

Gatewayed, "Open" Server

Example 4, above, shows how information transporter **14**, as well as server **22** can remain simple yet provide a highly general and extensible service. In that example, server **22** provides the functionality of a general-purpose transaction gateway or interface to an external function processor. In this particular case, the external function processor gatewayed by server **22** via vendor link **50**, is the merchant's order processing system, which receives the order, determines its disposition, and responds with order status information which is relayed back to server **22** for return to the customer as a response object in accord with protocols **38** and **44**. The user need not be aware of such complexities, nor do the client transport components **14** of the inventive product need to be aware of, or provide information for remote routing via vendor link **50**. Only the server **22** needs this information, and server **22** needs only to know that send objects with names that fall within a specified class for a specified product ID, must be forwarded to a specified external processor, and that the corresponding responses from that processor must be routed back to an originating client as response objects. Thus the inventive information transport component **14**, by virtue of its simplicity has general applicability and many uses, as described herein and as will further be apparent to those skilled in the art.

In implementing an ordering service using the inventive information transport component **14**, order and response objects are preferably formatted by the containing information product **12** to be consistent with existing or future electronic data interchange (EDI) standards which define protocols and formats for data interchange between customers and vendors. The information transport component **14** and the server protocol **44** provide the low-level EDI transport functions and are independent of object content defined by higher layers of the EDI protocol. Preferably, the server has added routing layer information to move objects to and from the external processor.

To provide a suitable EDI-compatible function, server **22** can be programmed with such higher layer EDI routing data for its exchanges with the merchant's external processor. Employing such a gatewayed system, a single EDI network connection can be used to connect the server **22** to a large number of different merchant processors anywhere in the world, across wide area networks and links between same, for example Internet.

This concept of an "open" server, providing a gatewayed pathway for information objects to travel between a wide base of users and one or more remote vendors or other object sources is greatly facilitated, or enabled, by employment of the inventive transporter **14** which effectively provides a protocol translation function enabling a simple information transport service to be offered which is easy and economical to use, both for the end user and the vendor or information supplier. Such a transport service compares favorably, for its intended information transport purposes with broader function and more complex of full online services, such as COMPUSERVE (trademark), and the like, described hereinabove.

Further Embodiments with Broadcast, Subscription Delivery and On-demand Capabilities

Receipt of broadcast data: As an alternative to modem-based wireline or wireless calling to a server and requesting data objects, the information transporter system of this invention can be beneficially employed in a broadcast information distribution system wherein data information objects are contained within a broadcast data stream with recipient communications devices tuned to identify and receive from the broadcast specific data elements to which they are entitled. On the Internet, such broadcasting to a selected group of recipients is called "multicasting".

Broadcasting can be airwave broadcasting via satellite, FM, or TV subchannels in the manner, for example, used by Mainstream Data Ltd. for the broadcast of news wires. Alternatively, the broadcast data stream may be cable or line transmitted, for example, over cable television systems. Minor extensions to API's **40** and **42** could accommodate such a facility. A modified setup function could alert a user's receiving communications device to watch for receipt of data objects identified as relating to the original or containing information product, and to capture and hold identified objects in temporary storage. A schedule transport function can then be set to fetch the received data objects from temporary storage and prepare them for use.

Subscription delivery: Although the invention has been described as being particularly applicable to the solution of problems arising in distributing updates of original or previously purchased or delivered electronic information products, those skilled in the art will appreciate that many of the benefits of the invention can be obtained without any initial information content being delivered to the user with the original product. The user could simply receive the information transporter **14** and all product information could be received subsequently, after installing the information transporter **14**, in the form of fetch objects transmitted from a remote server or other suitable source. For example, a newsletter service could provide a disk with the transporter and a user interface, but with no initial information content. As the transporter **14**, operating at the user stations, automatically fetches new issues according to the newsletter schedule, the information is, in effect, pushed down a channel from the distribution server for delivery to the base of subscribing users.

Information-on-demand services: In another embodiment, providing an information product on demand service, vendors can freely distribute a novel electronic marketing product comprising a transporter on diskette, along with a simple user interface and a catalog of information product items available from the vendor, without including the products themselves. Such an electronic marketing product could be distributed through the mail, as a magazine insert giveaway, or through any other suitable marketing medium. The transporter could be activated at any time by the user to call in and fetch a cataloged product, as well as a current catalog, possibly after sending a credit card order form, or the product price could be paid to the vendor by obtaining the product from a 900 number providing vendor reimbursement from the telephone network.

Open Architecture Online Service Access

In a further aspect, the invention provides an information transport component **14** that functions as universal or generic client interface software, enabling a user client to

work with any one or more of many online server-based information distribution services.

Many online information distribution services used to disseminate electronic publications comprise intelligent user interfaces which employ a client component running on a customer's personal computer (PC) to communicate with a central server facility operated by the online service, by means of a proprietary protocol. The client interface packages are proprietary to a particular online service.

Prospective publishers wishing to offer electronic products online, contract with online service providers to enable customers to use the online service's client software to access the publisher's material and related online communications services (bulletin boards, etc.) on the services' servers. The publisher is limited to using the presentation facilities provided by the user interface in the online service's client software. This limitation impedes migration of publisher offerings and makes it difficult for either a customer or a publisher to swing information transport component **14** access from one service provider to another because each service requires its own software package.

Third party interface developers cannot contribute to such online interfaces for a publisher without the cooperation of the online service provider which may be difficult or impossible to obtain. Accordingly, only limited user interfaces with moderate sophistication and variety can be offered.

Accordingly in another aspect, to provide open architecture online service communication, the inventive information transport component **14** can be embodied as a flexible client interface which can be actuated to operate with any one of a number of online services by providing a generic client interface foundation API (application program interface) combined with a set of translators and protocol drivers capable of communicating the user's functional requests to any one of a set of online services, using their corresponding proprietary protocols.

In this aspect the invention permits publishers to develop highly sophisticated and individualized user interfaces independently of the limitations of the online service providers' capabilities. Such enhanced user interfaces are attractive to publishers seeking differentiation of their products by providing an appealing individualized interface with a signature look and feel. In contrast, online service providers seeking to economically carry content from many publishers provide generic interfaces acceptable to all.

By incorporating operational translators for a number of online service protocols, which translators fully adhere to the detailed specifications of each protocol, a multiservice capability can be provided.

Online services generally provide similar types of services with nearly standard functions and similar user interfaces. Major service types include bulletin board, chat, electronic mail, document browsing, and database search. Use of creative typography, layout, graphics, and other artistic elements to offer the presentation quality and variety typical of print media is desired by publishers using this medium.

The invention facilitates this end by providing open development platforms for development of advanced interfaces while shielding developers from the complex details of communication with an online server. The shielding is accomplished by providing an API which supports communications service requests at a simple functional request level.

Referring to FIG. **3**, multiple targeted online services **80**, can be accessed by a client interface **82** comprising any of multiple graphical user interfaces **84** driving a generic API **86** which works with plug-in translator/communicator modules **88** which are provided to communicate one to each targeted online service **80**. Modules **88** mimic the online service's protocols, so as to be essentially indistinguishable from the proprietary interface s normally used. A communications manager **90** receives input from API **86** and outputs through protocol mapper **92** which selects the appropriate protocol.

In this embodiment, for use with full-function online services, the functions of API **86** and protocol **88** are extended to support extended, open-ended interactive sessions and the more varied client-server interaction needs of session-oriented interactive online applications such as bulletin board posting and browsing, online chat, electronic mail, database and menu browsing, and database search.

Similarly, in the aspect shown in FIG. **3**, the invention can be provided with the same kind of additional flexibility with regard to the user's connection to server **22** as the invention can provide for more basic fetch and send functions. While the inventive client server protocol **38** and **44** is particularly suited to the functions described, other existing or future services and corresponding protocols could be used, if necessary with adaptation, to provide workable services for use in conjunction with transport component **14**. Such use may require modification of communications module **36** and protocol **38** by the addition of a protocol mapper **92** and appropriate server protocol plug-in **88** to communicate to an alternative server.

In either case, such added flexibility in use of the inventive product increases a publisher's choices in selecting server and network facilities through which to distribute information products, and enables the publisher to offer fully customized user interfaces for use with multiple, or any one of multiple server and network services which do not provide for such customization. In this embodiment of the inventive transport component, a containing product can offer a unique custom interface and provide for access to additional information products from such varied source facilities as the Internet, full function online services, emerging groupware network services, conventional bulletin board systems, and future network services using wireless or cable television technology.

While the invention can provide a flexible, generic API, in some circumstances, an existing third-party API designed for use with a single specific online service can be combined with an embedded transporter and server protocol mapper to allow products designed to use the third-party API to employ any of multiple servers for distribution, avoiding commercial distribution restraints associated with that API, for example use of a particular server.

The inventive protocol mapper **92** can insulate a containing information product from the variations among such services, and can allow a single such information product to be transported through a variety of such services, and to later be moved to other such services by simply selecting an alternative protocol mapper. Multiple such protocol mappers can be packaged within a given information product to permit alternatives to be selected by the end-user from a list. Thus the invention further permits information products and related UI/DB authoring tools to be service-independent and neutral.

FIG. **4** provides an overview of the use of the inventive client interface accessing multiple publications via multiple remote online services, as well as multiple locally mounted data sources and storing additional retrieved data locally.

Enhancements can enable a publisher's service to provide integrated, seamless access to content distributed over sev-

eral different online services; to seamlessly combine access to both online and local CD-ROM-based content; and to coexist with and share resources with other publishers' services on the user's PC.

In summary, the invention provides, in this aspect, a simple, easy-to-use multi-protocol capability that enables an electronic information object to be transported from a publisher to a wide base of users by any one of a number of online services, without sacrificing individual product identity.

Recursive Updating of the Transporter

Another application of the inventive information transport product, or transporter, is a recursive use to update itself, in the same manner that the transporter can update a containing information product. This method can be useful in a variety of ways, including to upgrade the transporter by the addition of new protocol components, new compression techniques, or new network access methods.

An important class of such self-updates is to provide added flexibility in specifying network access procedures. For example, the user setup routine could be extended into a two stage process. In a first stage, each user's transporter calls in to a common pre-set phone number, in order to fetch a second phone number selected according to the user's particular product, location, or some other parameter. The second phone number, or other address, can then placed in the setup as an update, to be used in subsequent transport operations.

This two-stage method can provide efficient use of a single pre-set toll-free 800 number for an initial call from any number of different products, which initial call yields a second number corresponding to a specific Product ID, which number is used for subsequent calls.

In an advantageous embodiment, the second number is not toll free and may include vendor charges, in the manner of a 900 number. This arrangement enables a system in which users do not pay for initial setup calls (and any failed connections which might result from initial setup problems), but do pay long-distance toll charges, and per call vendor fees if the publisher so desires, for subsequent product information transport from the second number. This two-number process can be carried out without requiring any phone number entry or selection by the user. Additionally, the second number can readily be changed whenever desired by the publisher, even after product discs have been shipped.

User's Station

References herein to a user's station, workstation, computer or terminal will be understood to embrace any "information appliance" or intelligent device having the basic computer-like functions of programmed logic, storage and presentation, or having the ability to support an operating system for managing user input-output with a processor, including intelligent cable television controllers, video game players, information kiosks, wired and wireless personal communicators, and even system controllers such as automotive computers.

Benefits Provided by the Invention

Employing the novel information transport component **14** interacting with remote server **22** through communications protocols **38** and **44**, the invention enables the following advantageous objectives and other benefits to be achieved:

i) simple and easy execution of one or more fetch or send transactions to or from a remote server, by an ordinary, unskilled user with no human interaction at either end being necessary after initiation;

ii) automated transport of predefined information objects between client and server in a closed-ended fashion, without burdening a client-based user with complex routing logic; and

iii) creation of an economic, easy-to-use, function-specific, self-contained information transport component **14** software module suitable for mass distribution in a containing information product.

The preferred use of an object manifest in a transport control mechanism which includes transporting the object manifest between client user and server, and referencing the object manifest by user fetch-send protocol **38** and server fetch-send protocol **44** facilitates achievement of the following additional objectives:

iv) simple, tight-knit control of the communication process and of error handling; and

v) creation of a transport control mechanism, and thence of an information transport component **14**, which operates smoothly and transparently to the user and independently of the information object content or of the nature of the application.

The invention thus provides an information transport software component which can be employed to transport a wide variety of data objects or applications and can be easily incorporated in many different information products to provide multiple novel containing information products **12** with built-in automated updatability or upgradability executable at an appropriate time by simple, user-menu selection or automatically.

Further Benefits

In addition to the benefits of a powerful and efficient information transport method, use of a standard, formalized transporter, its API, and client-server protocol, pursuant to the teachings of the invention disclosed herein, can provide any or all of the following significant benefits to users, information product vendors, application vendors, service providers, tool vendors or others:

vi) use of a standardized facility to perform a well defined function in a known way (with available implementations for a varied and expanding set of hardware and software platforms);

vii) reliance on a standardized facility that can be extensively tested and proven reliable across a wide variety of equipment and conditions;

viii) reduced need for information product developers (and users, and user interface/database search software vendors) to know and understand the complexities (and rapid evolution) of data communications;

ix) ability to build a single functional interface that can smoothly employ a dynamically expanding variety of communications facilities and technologies;

x) ability to obtain operations and user support services relating to the difficult task of managing a server and its communications with large numbers of end-users;

xi) user-recognition of the novel information transport facility across a range of unrelated products, establishing a positive brand cachet benefiting users and vendors alike;

xii) ability to package the transporter facility with other tools, such as a UI(user interface) and database search capability to extend the value of those tools economically and with the ability to gain the benefits described above; and

xiii) control of communications costs and failures by elimination of human intervention, with its attendant time-consuming delays and errors, from the period during which the user's station is connected in real time communication with remote server **22**.

Stated succinctly, by having the novel information transport component rely entirely on a containing information product for all user interface and information presentation functions, there need be no restrictions on the creativity of the containing product imposed by the needs of a third party communications product. Thus the containing information product can present transport functions with any desired look and feel.

Another advantage of the information transport system of the invention is the avoidance of difficult or complex navigation tasks, and the use of simple direct dial communications which are suitable for sessions that are short and infrequent. The inventive information transport products described herein are consistent with or readily adaptable to the needs of many publishers of a diversity of materials, which needs are commonly centered on discrete products and content.

A further advantage of the invention, from the point of view of publishers, is that because the call is customer initiated, the customer pays transport costs (telephone line charges), simplifying costing for the publisher who avoids having to figure shipment or other transportation costs before sale and build these costs into the price of the product or update.

The inventive approach to mass distribution of electronic information products described herein can also provide advantages in high-value environments such as those of Counterpoint Publishing's Federal Register products cited hereinabove, providing a more seamless integration of the fetching of updates received via modem (and selected and extracted by the user from the "Daily Federal Register") with the original product on CD-ROM, the "CD Federal Register". Product installation can be simplified, and a separate user invocation of, and interface to, a general purpose communications package can be avoided. In addition, by employing the user's pre-existing modem and avoiding need for a general purpose communications product license, significant cost savings can be obtained.

The better to comprehend its possible applications and enhancements, embodiments of the invention can be grouped in four levels, as a follows.

Level zero A novel basic transport function embeddable in any of a range of electronic information products to provide economical unattended updates.

Level one Basic transporter **14** incorporating API's **40** and **42** adds a powerful new capability to be used with an electronic information product's custom user interface and database management facility for seamless integration of an update with an original product. other options can integrate with relevant third-party packages such as authoring packages.

Level one (Server enhanced) Adds server operation and user support features enabling publishers to outsource tasks which may be difficult or unfamiliar to them.

Level Two Adds optional translation or use of alternative server protocols enabling an embeddable transporter product to work with many different servers or services including, for example, standard BBS's, Internet servers, and special transport services such as those offered or proposed by communications providers such as AT&T, MCI, Compuserve, America Online and cable television systems.

Level Three Adds a full online service user interface API with correspondingly enhanced client-server protocols to provide for full-function online service sessions with user interface control and with ability to work with a range of online services, providing a publisher with flexibility in their use of existing and emerging services.

While an illustrative embodiment of the invention has been described above, it is, of course, understood that various modifications will be apparent to those of ordinary skill in the art. Such modifications are within the spirit and scope of the invention, which is limited and defined only by the appended claims.

What is claimed is:

1. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein a higher level software entity can be invoked to modify the object manifest, and

wherein the higher level software entity comprises a viewer for at least one content type available on the communications network, the content type being selected from the group consisting of multimedia formats, video formats, sound formats and hypertext markup language ("HTML").

2. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein a higher level software entity can be invoked to modify the object manifest,

wherein the higher level software entity comprises a viewer for at least one content type available on the communications network, the content type being selected from the group consisting of multimedia formats, video formats, sound formats and hypertext markup language ("HTML"), and

wherein the communications network is the Internet.

3. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein:

i) the communications network is a broadcast network comprising multiple user stations each provided with the information transporter;

ii) at least one of the remote sources broadcasts a data stream across the network for receipt by the user stations;

iii) the object manifest at each user station defines data stream content elements for receipt by the user station; and

iv) a higher level software entity can be invoked to modify the object manifest.

4. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported,

wherein:

i) the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by at least one of the remote sources;

ii) the at least one remote source has, for each user station, an object manifest received across the network from the user station and specifying user station identification information;

iii) each user station repeatedly receives objects transported by the at least one remote source; and

iv) a higher level software entity can be invoked to modify the object manifest.

5. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported,

wherein:

i) the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by at least one of the remote sources;

ii) the at least one remote source has, for each user station, an object manifest received across the network from the user station and specifying user station identification information;

iii) each user station repeatedly receives objects transported by the at least one remote source;

iv) a higher level software entity can be invoked to modify the object manifest; and

v) the object manifest received at the remote source specifies user-desired content and the information objects transported by the remote source to the user station are selected according to the user-desired content specification.

6. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported,

wherein:

i) the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by at least one of the remote sources;

ii) the at least one remote source has, for each user station, an object manifest received across the network from the user station and specifying user station identification information;

iii) each user station repeatedly receives objects transported by the at least one remote source;

iv) a higher level software entity can be invoked to modify the object manifest;

v) the object manifest received at the remote source specifies user-desired content and the information objects transported by the remote source to the user station are selected according to the user-desired content specification; and

vi) the user-desired content specification comprises a generic or an alias name to request a latest installment, version or update.

7. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein:

i) the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by at least one of the remote sources;

ii) the at least one of the remote sources has, for each user station, an object manifest received across the network from the user station and specifying user station identification information;

iii) each object manifest contains user-specified information object selections; and

iv) each user station transporter is scheduled to communicate repeatedly and automatically with the at least one of the remote sources and fetch information objects meeting the user-specified information object selections.

8. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein:

i) the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by at least one of the remote sources;

ii) the at least one of the remote sources has, for each user station, an object manifest received across the network from the user station and comprising user-specified information object selections; and

iii) each user station transporter can fetch or receive a response object from the one of the remote sources providing the user-specified information object selections; and

iv) a higher level software entity can be invoked to modify the object manifest.

9. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein a higher level software entity can be invoked to modify the object manifest, and

wherein the information transporter is embedded in a containing information product, the transporter functionality being activatable via the information product.

10. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein a higher level software entity can be invoked to modify the object manifest,

wherein the information transporter is embedded in a containing information product, the transporter functionality being activatable via the information product, and

wherein the containing information product is selected from the group consisting of self-updated software products, self-updating database products, CD-ROM resident products, online hybrid products, Internet access products, offline Internet access products, mobile Internet access products, short-session Internet access products, or intelligent appliance products.

11. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein a higher level software entity can be invoked to modify the object manifest, and

wherein the transport control module specifies object processing actions required to integrate an object into one of an application and an information product on the user station subsequent to transport.

12. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein a higher level software entity can be invoked to modify the object manifest, and

wherein the manifest list is mobile and transportable in the transport session, moving in a predetermined direction between the source station and the user station to request at least one information object to be sent in the other direction between the source station and the user station.

13. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein:

the transport control module uses an object manifest comprising at least one of a send object list, and a fetch object list, and

the user station includes a user interface provided by a vendor associated with the source, and

the object manifest is created under control of the user interface from choices supplied by the vendor.

14. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein:

a higher level software entity can be invoked to modify the object manifest;

the user station is capable of executing multiple communications protocols;

the transport control module is responsive to a protocol selection code; and

the send object list comprises one or more object list elements selected from the group consisting of object action codes specifying source station actions required, object names, object sizes and response object size.

15. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein:

a higher level software entity can be invoked to modify the object manifest;

the transport control module is responsive to a completed object manifest having codes to convey the status of the transport operation or to provide for transport of additional information objects, or both; and

for a send operation in which an information object is transported from the user station to the source, the completed object manifest comprises one or more manifest elements selected from the group consisting of send object additional information, object acceptance codes returned by the source, time of object acceptance codes, response object names and a completion status code to terminate the send operation and return control.

16. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein:

a higher level software entity can be invoked to modify the object manifest;

the transport control module is responsive to a completed object manifest having codes to convey the status of the transport operation or to provide for transport of additional information objects, or both;

for a send operation in which an information object is transported from the user station to the source, the completed object manifest comprises one or more manifest elements selected from the group consisting of send object additional information, object acceptance codes returned by the source, time of object acceptance codes, response object names and a completion status code to terminate the send operation and return control; and

the completed object manifest further comprises polling indicator codes enabling polling of the user station by the source for readiness to perform additional transport operations.

17. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein:

a higher level software entity can be invoked to modify the object manifest

the transport control module is responsive to a completed object manifest having codes to convey the status of the transport operation or to provide for transport of additional information objects, or both;

for a send operation in which an information object is transported from the user station to the source, the completed object manifest comprises one or more manifest elements selected from the group consisting of send object additional information, object acceptance codes returned by the source, time of object acceptance codes, response object names and a completion status code to terminate the send operation and return control; and

the completed object manifest further comprises scheduled update indicator codes enabling scheduled fetching of updates by the user station from the source.

18. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein a higher level software entity can be invoked to modify the object manifest, and

wherein:

the transport control module comprises a transport software component embeddable in a vendor-provided information product;

the vendor provides update objects from a selected source; and

the transport software component is separately suppliable to multiple vendors of respective information products.

19. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a communications module which effects the fetching or sending of information objects across the network between at least one of the remote sources and persistent storage at the user station; and

(b) a transport control module which controls transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) an object manifest specifying at least one information object to be transported;

wherein:

a higher level software entity can be invoked to modify the object manifest;

the transport control module comprises a transport software component embeddable in a vendor-provided information product;

the vendor provides update objects from a selected source;

the transport software component is separately suppliable to multiple vendors of respective information products; and

the transport software component further comprises a vendor-related user interface permitting specification of transport objects in the object manifest.

20. A method of controlling transport of information objects between persistent storage at a user station and a remote information object source on a communications network using an information transporter comprising a communications module for sending and receiving information objects on the network and wherein the method comprises:

(a) communicating object transport specifications, including a source address for the remote information object source, between the information transporter and a higher level software entity employing an object manifest listing at least one information object to be transported;

(b) activating the communications module to transport the at least one information object to or from the source address, in accordance with the object manifest; and

(c) scheduling the transporter to communicate repeatedly and automatically with the remote information object source and transport information objects,

wherein:

the communications network is a broadcast network comprising multiple user stations each provided with the information transporter; and

the remote information object source broadcasts a data stream across the network for receipt by the user stations, the method further comprising:

(d) receiving at each user station data stream content elements defined by specifications in the object manifest.

21. A method of controlling transport of information objects between persistent storage at a user station and a remote information object source on a communications network using an information transporter comprising a communications module for sending and receiving information objects on the network and wherein the method comprises:

(a) communicating object transport specifications, including a source address for the remote information object source, between the information transporter and a higher level software entity employing an object manifest listing at least one information object to be transported;

(b) activating the communications module to transport the at least one information object to or from the source, address in accordance with the object manifest; and

(c) scheduling the transporter to communicate repeatedly and automatically with the remote information object source and transport information objects,

wherein the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by the remote information object source, the method further comprising:

(d) sending to the remote information object source from each user station an object manifest specifying user station identification information; and

(e) repeatedly transporting information objects to each user station from the remote information object source.

22. A method of controlling transport of information objects between persistent storage at a user station and a remote information object source on a communications network using an information transporter comprising a communications module for sending and receiving information objects on the network and wherein the method comprises:

(a) communicating object transport specifications, including a source address for the remote information object source, between the information transporter and a higher level software entity employing an object manifest listing at least one information object to be transported;

(b) activating the communications module to transport the at least one information object to or from the source address, in accordance with the object manifest; and

(c) scheduling the transporter to communicate repeatedly and automatically with the remote information object source and transport information objects,

wherein the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by the remote information object source, the method further comprising:

(d) sending to the remote information object source from each user station an object manifest specifying user station identification information; and

(e) repeatedly transporting information objects to each user station from the remote information object source, wherein the object manifest received at the remote information object source specifies user-desired content with a generic or an alias name and wherein the method further comprises:

(f) the remote information object source sending to the user station the latest installment, version or update information objects selected according to the generic or alias name.

23. A method of controlling transport of information objects between persistent storage at a user station and an information object remote source on a communications network using an information transporter comprising a communications module for sending and receiving information objects on the network, the method comprising:

(a) communicating object transport specifications, including a source address for the remote source, between the information transporter and a higher level software entity employing an object manifest listing an information object to be transported;

(b) activating the communications module to transport the information object to or from the source address, in accordance with the object manifest,

wherein the communications network comprises a group of user stations each provided with the information transporter, the object manifest at each user station contains source-originated information object specifications and each user station is a client station of an information object distribution service provided by the remote source and wherein the method further comprises:

(c) scheduling each transporter to communicate repeatedly and automatically with the remote source and transport information objects meeting the source-originated specifications.

24. A method of controlling transport of information objects between persistent storage at a user station and an information object remote source on a communications network using an information transporter comprising a communications module for sending and receiving information objects on the network, the method comprising:

(a) communicating object transport specifications, including a source address for the remote source, between the information transporter and a higher level software entity employing an object manifest listing at least one information object to be transported;

(b) activating the communications module to transport the at least one information object to or from the source address, in accordance with the object manifest,

wherein the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by the remote source, the method further comprising:

(c) sending to the remote source, from each user station, the object manifest comprising user-specified information object selections; and

(d) scheduling each transporter to communicate repeatedly and automatically with the remote source and transport information objects meeting the user specifications.

25. A method of controlling transport of information objects between persistent storage at a user station and an information object remote source on a communications network using an information transporter comprising a communications module for sending and receiving information objects on the network, the method comprising:

(a) communicating object transport specifications, including a source address for the remote source, between the information transporter and a higher level software entity employing an object manifest listing at least one information object to be transported;

(b) activating the communications module to transport the at least one information object to or from the source address, in accordance with the object manifest,

wherein the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by at least one of the remote sources, the method further comprising:

c) fetching a features directory listings features available at the remote source from the remote source to each user station;

d) building, at each user station, an object manifest containing selected feature entries obtained from the fetched features directory; and

e) scheduling each user station transporter to communicate repeatedly and automatically with the remote

source and transport information objects specified by the selected features entries in the object manifest.

26. A method of controlling transport of information objects between persistent storage at a user station and a remote information object source on a communications network using an information transporter comprising a communications module for sending and receiving information objects on the network and wherein the method comprises:

(a) communicating object transport specifications, including a source address for the remote information object source, between the information transporter and a higher level software entity employing an object manifest listing at least one information object to be transported;

(b) activating the communications module to transport the at least one information object to or from the source address, in accordance with the object manifest; and

(c) scheduling the transporter to communicate repeatedly and automatically with the remote information object source and transport information objects,

wherein the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by the remote source, the method further comprising:

(d) sending to the remote source, from each user station, an object manifest comprising user specified information object selections; and

(e) using each user station transporter to fetch or receive a response object from the remote source providing the user-specified information object selection.

27. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a separable communications module which selectively fetches or transports information objects across the network between at least one of the remote sources and the user station; and

(b) a transport control module which controls the transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) a user-modifiable object manifest specifying at least one information object to be transported,

wherein:

i) the communications network is a broadcast network comprising multiple user stations each provided with the information transporter;

ii) the at least one of the remote sources broadcasts a data stream across the network for receipt by the user stations; and

iii) the object manifest at each user station defines data stream content elements for receipt by the user station.

28. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a separable communications module which selectively fetches or transports information objects across the network between at least one of the remote sources and the user station; and

(b) a transport control module which controls the transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) a user-modifiable object manifest specifying at least one information object to be transported,

wherein:

i) the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by the at least one of the remote sources;

ii) the at least one remote source has, for each user station, an object manifest received across the network from the user station and specifying user station identification information; and

iii) each user station repeatedly receives objects transported by the at least one remote source.

29. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a separable communications module which selectively fetches or transports information objects across the network between at least one of the remote sources and the user station; and

(b) a transport control module which controls the transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) a user-modifiable object manifest specifying at least one information object to be transported,

wherein:

i) the communications network comprises a group of user stations each provided with the information transporter and each being a client station of an information object distribution service provided by the at least one of the remote sources:

ii) the at least one of the remote sources has, for each user station, an object manifest received across the network from the user station and comprising user-specified information object selections; and

iii) each user station transporter can fetch or receive a response object from the one of the remote sources providing the user-specified information object selections.

30. An automated electronic information transporter located at a user station for controlling transport of information objects on a communications network providing access to multiple remote sources, the information transporter comprising:

(a) a separable communications module which selectively fetches or transports information objects across the network between at least one of the remote sources and the user station; and

(b) a transport control module which controls the transport of the information objects in accordance with:

i) a source address for the at least one remote source; and

ii) a user-modifiable object manifest specifying at least one information object to be transported,

wherein:

the information transporter is embedded in a containing information product; and

the information transporter functionality can be activated during operation of the information product.

* * * * *

APPENDIX F

U.S. Patent No. 6,199,082 to Ferrel *et al.*

US006199082B1

(12) **United States Patent**
Ferrel et al.

(10) **Patent No.: US 6,199,082 B1**
(45) **Date of Patent: *Mar. 6, 2001**

(54) **METHOD FOR DELIVERING SEPARATE DESIGN AND CONTENT IN A MULTIMEDIA PUBLISHING SYSTEM**

(75) Inventors: **Patrick J. Ferrel**, Seattle; **Robert F. Meyer**, Redmond; **Stephen J. Millet**, Seattle; **John P. Shewchuk**, Seattle; **Walter W. Smith**, Seattle, all of WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/503,343**

(22) Filed: **Jul. 17, 1995**

(51) **Int. Cl.**[7] **G06F 17/21**
(52) **U.S. Cl.** **707/522**; 707/515
(58) **Field of Search** 395/774, 776–779, 395/784, 788; 707/500, 522; 767/513, 501, 514–515; 709/203–219; 345/302

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,723,211		2/1988	Barker et al.	395/776
4,949,287		8/1990	Yamaguchi et al.	395/782
4,962,475		10/1990	Hernandez et al.	395/777
4,969,093	*	11/1990	Barker et al.	395/800
5,051,930		9/1991	Kuwabara et al.	395/778
5,181,162	*	1/1993	Smith et al.	395/792
5,339,392	*	8/1994	Risberg et al.	395/333
5,347,632		9/1994	Filepp et al.	395/200.09
5,491,785	*	2/1996	Robson et al.	395/507
5,491,820	*	2/1996	Belove et al.	395/603
5,557,722	*	9/1996	DeRose et al.	395/774
5,630,103	*	5/1997	Smith et al.	395/500
5,666,542	*	9/1997	Katai et al.	707/522

OTHER PUBLICATIONS

Simpson, Mastering Wordperfect for Windows, pp. 838–839, 1993.*

Bormann et al., "Standards for open document processing: current state and future developments", *Computer Networks/ and ISDN Systems*, v. 21, pp. 149–163, Jan. 1991.*

Gifford, "Polychannel systems for mass digital communication", *Comm of ACM*, v. 33, n. 2, p. 141 (11), Feb. 1990.*

Schlichter et al., "FolioPub: A Publication Management System", *IEEE Computer*, v. 21, n. 1, pp. 61–69, Jan. 1988.*

Nakano et al., An Interface/application Builder with Transmission Control Facility for SGML document Databases, Proceedings of the Intl. Conf. on Database and Expert Systems Applications (DEXA '91), pp. 41–46, Jan. 1991.*

Journalist User's Guide: Your Personalized Newspaper for CompuServe ®, PED Software Corp., pp. 1–111, Jan. 1994.*

Huser et al., "The Individualized Electronic Newpaper", GMD Report No. GMD–664, Jul. 1992.*

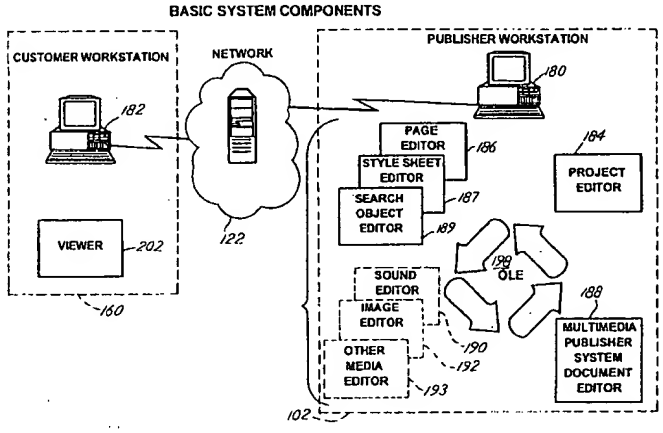
(List continued on next page.)

Primary Examiner—Joseph H. Feild
(74) *Attorney, Agent, or Firm*—Banner & Witcoff, Ltd.

(57) **ABSTRACT**

A multimedia publishing system where the format and content can be separated and uploaded to a server by a publisher. Usually, the format used by publishers remains reasonably constant over time, contrasted with the content which changes on a regular basis. As content changes on a regular basis, the publisher uploads only the new content to the server. When clients or customers access the server's content, the server downloads the format and content to the user's computer. Subsequent downloads of content transmits only the content since the format is cached on the customer's computer after the first download. If the publisher desires to change the format at a subsequent time, the next download of content by the customer downloads both the new layout format and the new content. This publication scheme minimizes the transmission of data in bandwidth limited environments.

29 Claims, 23 Drawing Sheets



OTHER PUBLICATIONS

Mendelson, Edward, "Share and Share Alike," Computer Shopper, Ziff Davis Publishing, Feb. 1995, pp. 518–521, 524, 526–527, 529.

Chao, Julie, "Adobe Systems Sees Cyberspace as a Brave New Market: Software Maker's New Program Will Help Users Navigate the Internet," The Wall Street Journal (Corporate Focus), May 15, 1995, p. B4.

Rupley, Sebastian, "Folio's On–Line Business Library," PC Magazine (Trends), May 16, 1995, p. 32.

"The Wall Street Journal Introduces the First Newspaper Published for a Circulation of One. Personal Journal," The Wall Street Journal, May 16, 1995.

Patrizio, Andy, "Internet: Sun ramps up hardware, software offerings," PC Week, May, 22, 1995, p. 3.

Barney, Doug, "On–line publishing: Publishers wary of Notes service," INFOWORLD, May 29, 1995, p. 8.

Leach, Norvin, "It's only in alpha, but testers say Java is hot," PC Week, May 29, 1995, p. 92.

* cited by examiner

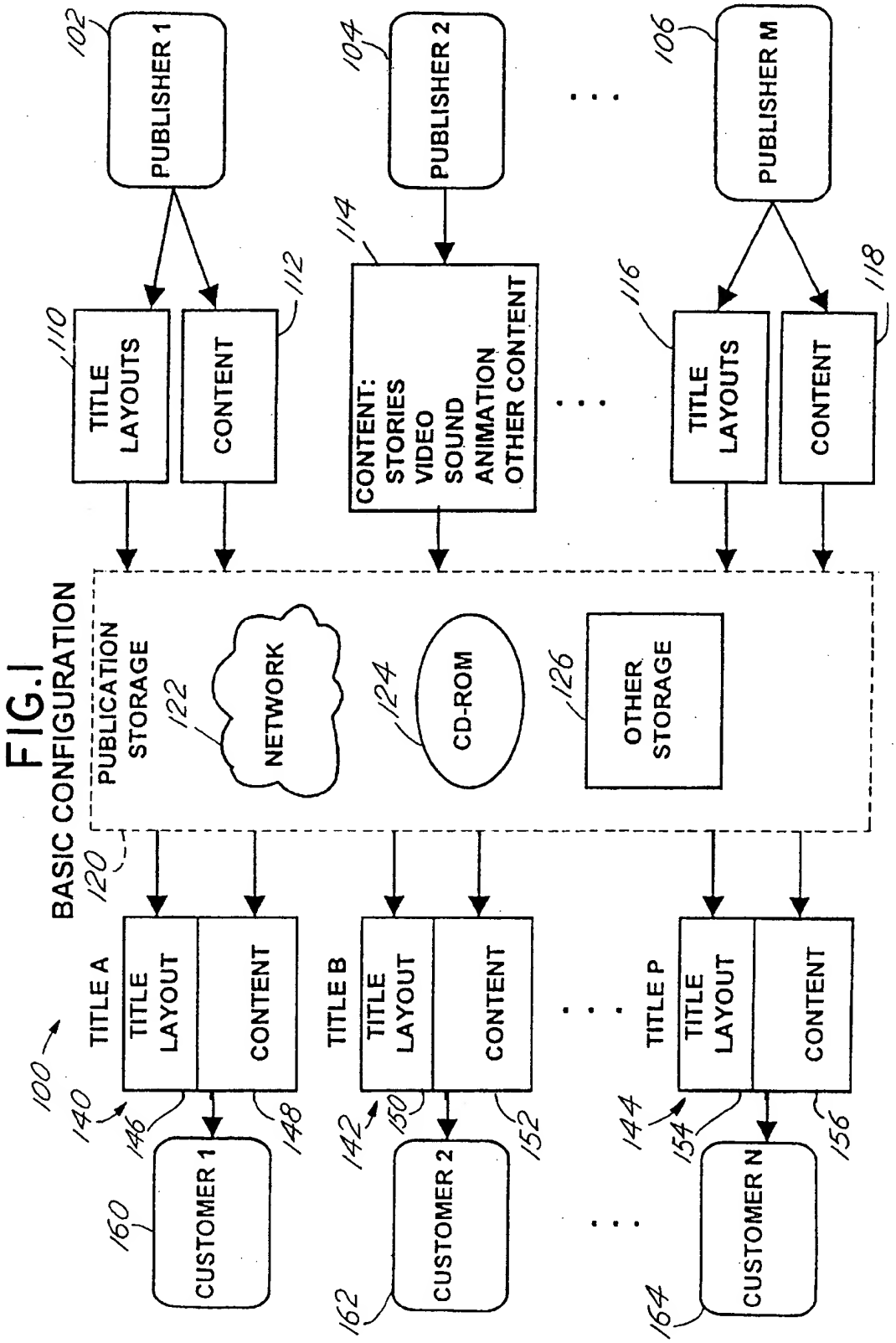

BASIC CONFIGURATION
100
PUBLISHER 1
102
PUBLISHER 2
104
PUBLISHER M
106
TITLE LAYOUTS
110
CONTENT
112
CONTENT: STORIES VIDEO SOUND ANIMATION OTHER CONTENT
114
TITLE LAYOUTS
116
CONTENT
118
PUBLICATION STORAGE
120
NETWORK
122
CD-ROM
124
OTHER STORAGE
126
TITLE A
TITLE LAYOUT
CONTENT
140
146
148
TITLE B
TITLE LAYOUT
CONTENT
142
150
152
TITLE P
TITLE LAYOUT
CONTENT
144
154
156
CUSTOMER 1
160
CUSTOMER 2
162
CUSTOMER N
164

FIG. 1

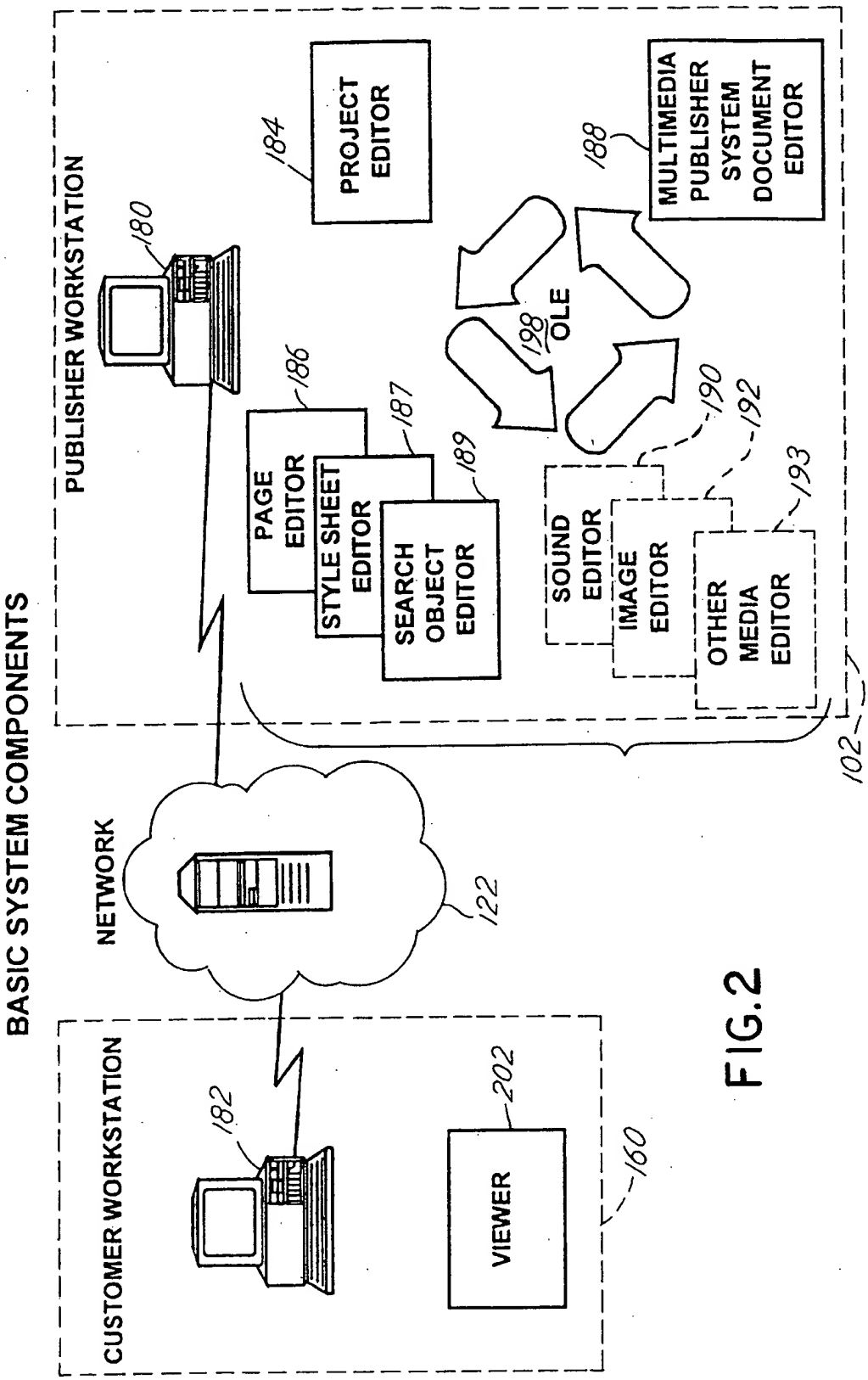
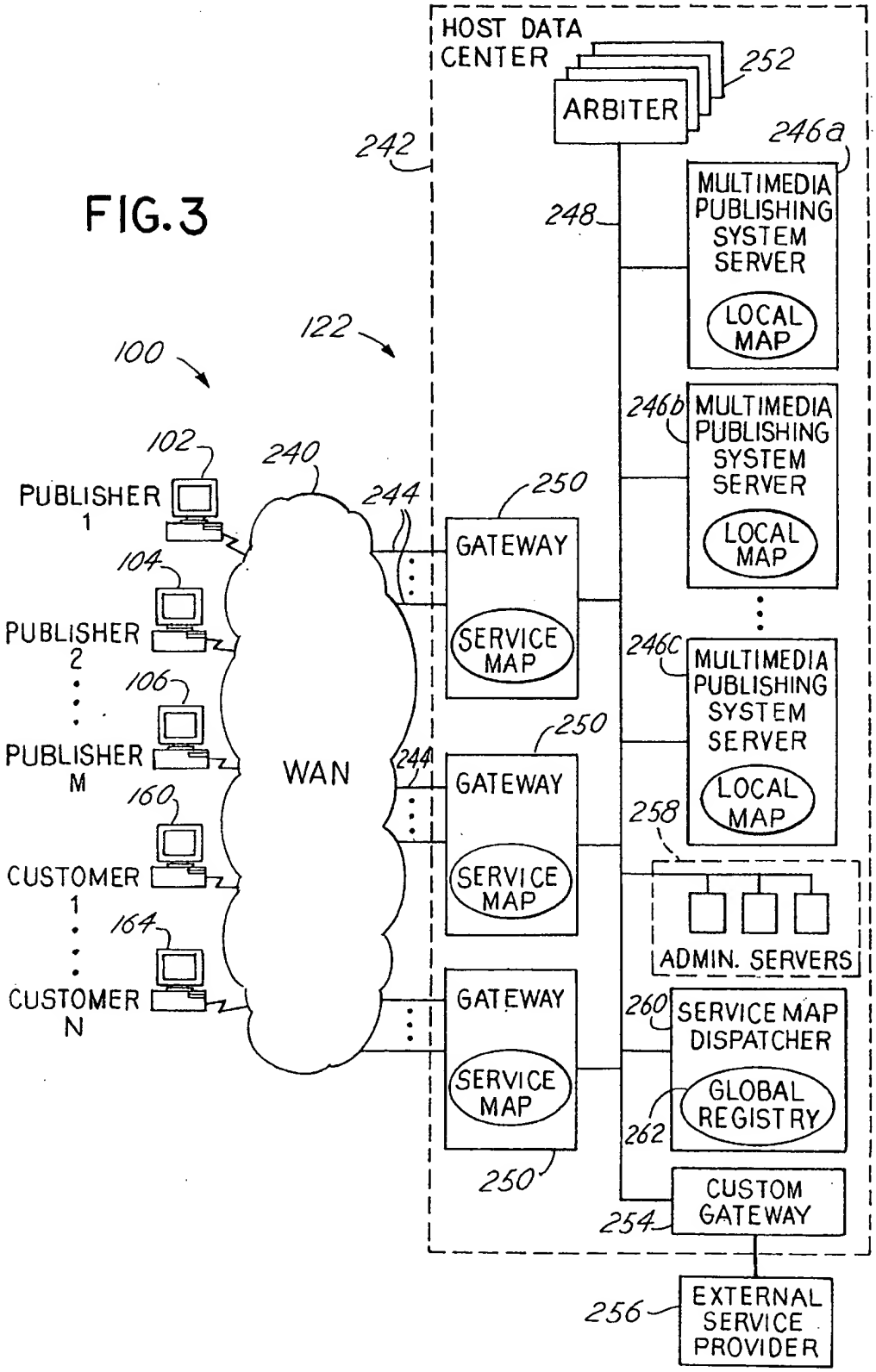

BASIC SYSTEM COMPONENTS
PUBLISHER WORKSTATION
180
184
PROJECT EDITOR
188
MULTIMEDIA PUBLISHER SYSTEM DOCUMENT EDITOR
198
OLE
186
187
189
PAGE EDITOR
STYLE SHEET EDITOR
SEARCH OBJECT EDITOR
190
192
193
SOUND EDITOR
IMAGE EDITOR
OTHER MEDIA EDITOR
102
NETWORK
122
CUSTOMER WORKSTATION
182
202
VIEWER
160

FIG.2

FIG.3


HOST DATA CENTER
252
ARBITER
242
246a
MULTIMEDIA PUBLISHING SYSTEM SERVER
LOCAL MAP
248
246b
MULTIMEDIA PUBLISHING SYSTEM SERVER
LOCAL MAP
246c
MULTIMEDIA PUBLISHING SYSTEM SERVER
LOCAL MAP
258
ADMIN. SERVERS
260
SERVICE MAP DISPATCHER
GLOBAL REGISTRY
262
CUSTOM GATEWAY
254
EXTERNAL SERVICE PROVIDER
256
122
100
102
PUBLISHER 1
104
PUBLISHER 2
106
PUBLISHER M
160
CUSTOMER 1
164
CUSTOMER N
240
WAN
244
250
GATEWAY
SERVICE MAP
GATEWAY
SERVICE MAP
GATEWAY
SERVICE MAP
250

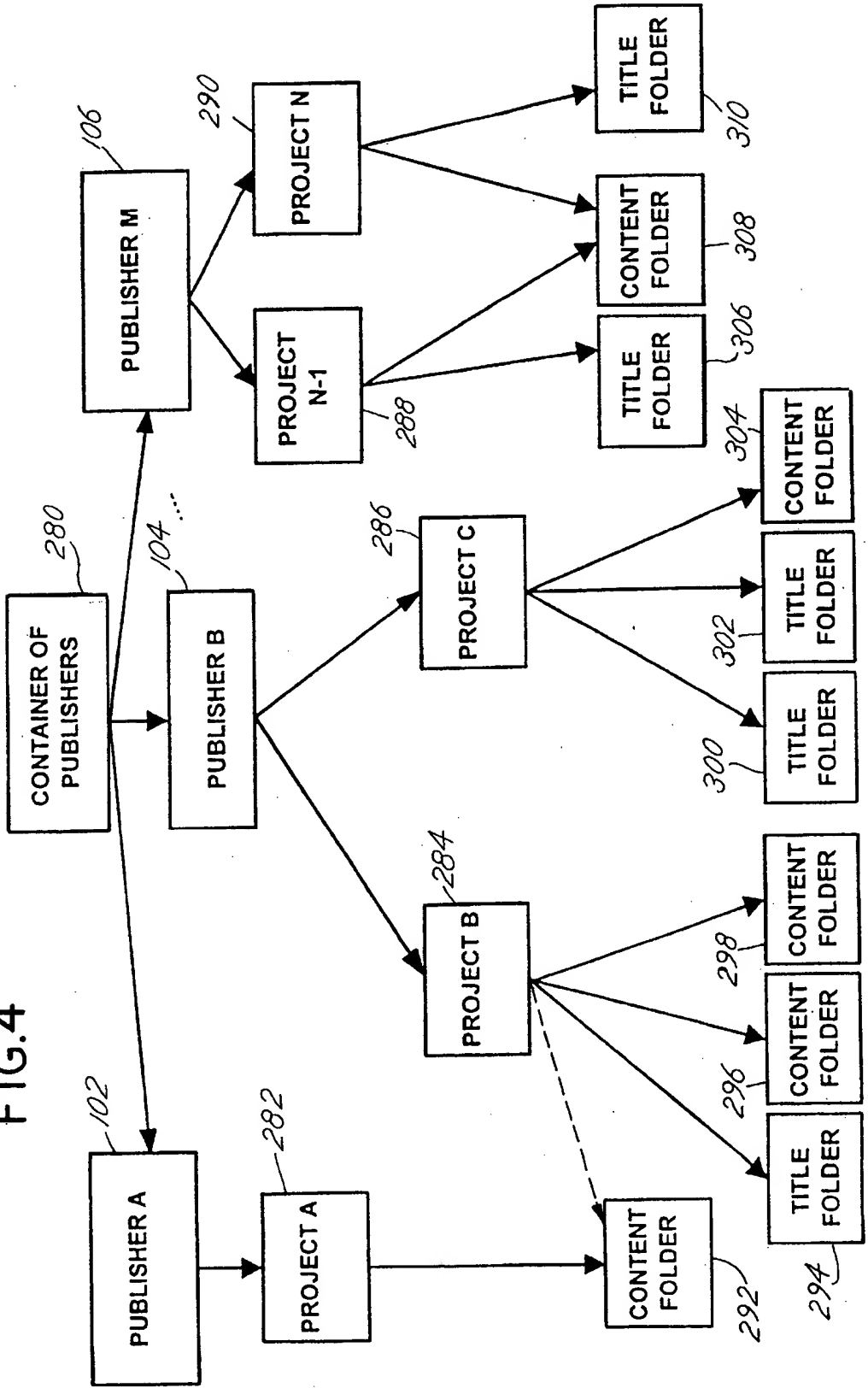

CONTAINER OF PUBLISHERS
280
PUBLISHER A
102
PUBLISHER B
104
PUBLISHER M
106
PROJECT A
282
PROJECT B
284
PROJECT C
286
PROJECT N-1
288
PROJECT N
290
CONTENT FOLDER
292
TITLE FOLDER
294
CONTENT FOLDER
296
CONTENT FOLDER
298
TITLE FOLDER
300
TITLE FOLDER
302
CONTENT FOLDER
304
TITLE FOLDER
306
CONTENT FOLDER
308
TITLE FOLDER
310

FIG.4

FIG. 5

SYSTEM OVERVIEW

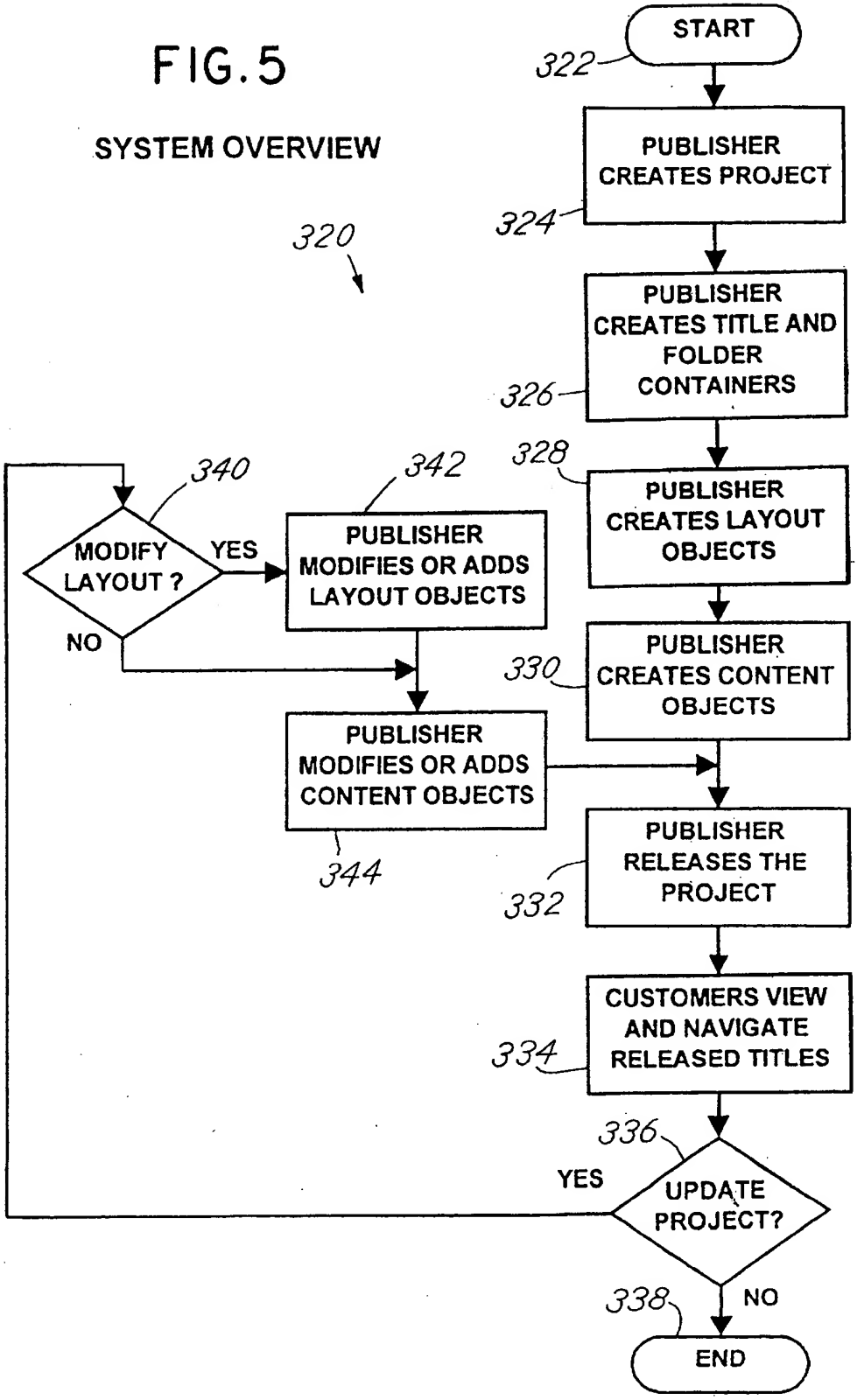
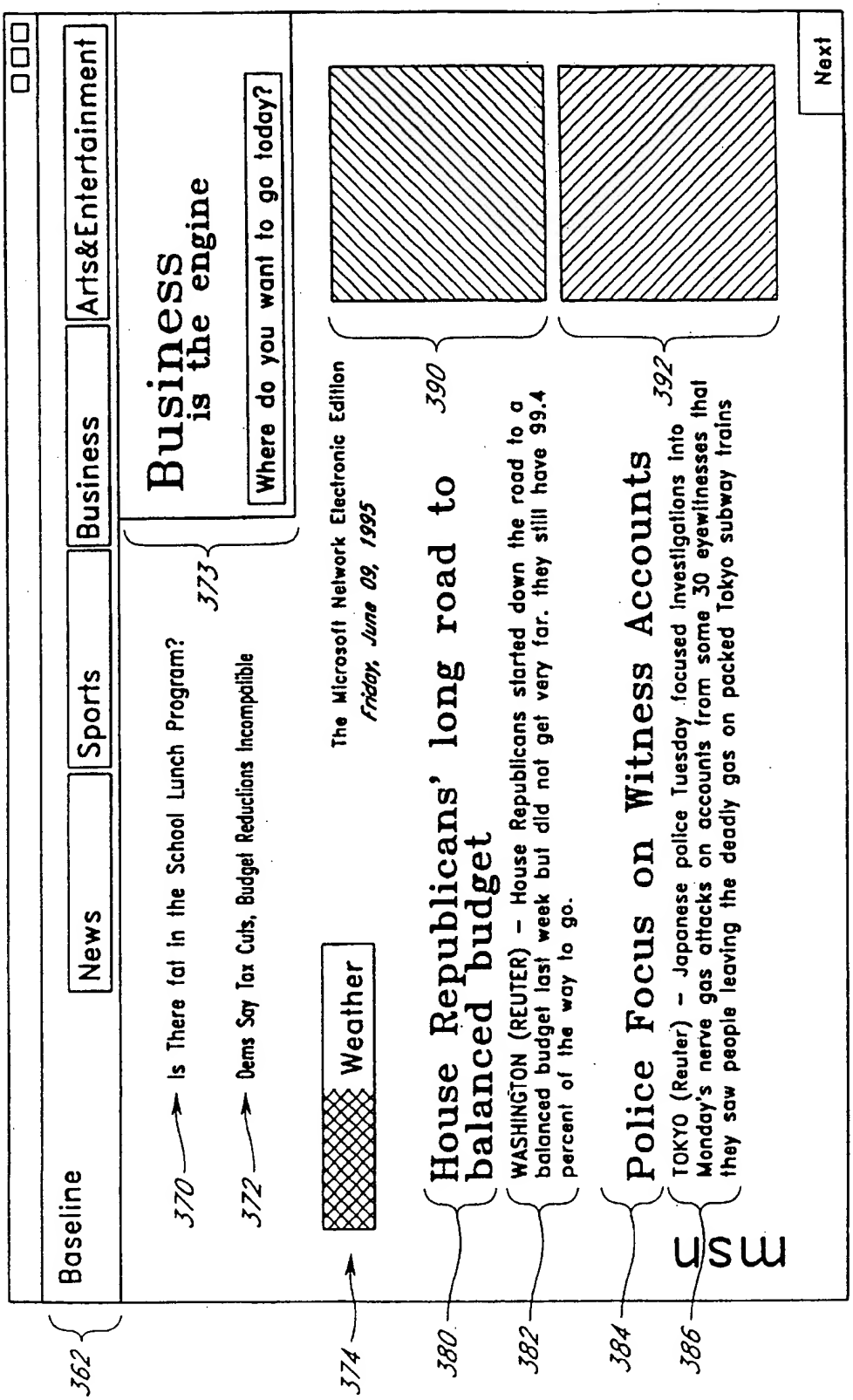


FIG.6


360
362
Baseline
News
Sports
Business
Arts&Entertainment
370
Is There fat in the School Lunch Program?
372
Dems Say Tax Cuts, Budget Reductions Incompatible
373
Business is the engine
Where do you want to go today?
374
Weather
The Microsoft Network Electronic Edition
Friday, June 09, 1995
380
House Republicans' long road to balanced budget
390
382
WASHINGTON (REUTER) - House Republicans started down the road to a balanced budget last week but did not get very far. they still have 99.4 percent of the way to go.
384
Police Focus on Witness Accounts
392
386
TOKYO (Reuter) - Japanese police Tuesday focused investigations into Monday's nerve gas attacks on accounts from some 30 eyewitnesses that they saw people leaving the deadly gas on packed Tokyo subway trains
Next

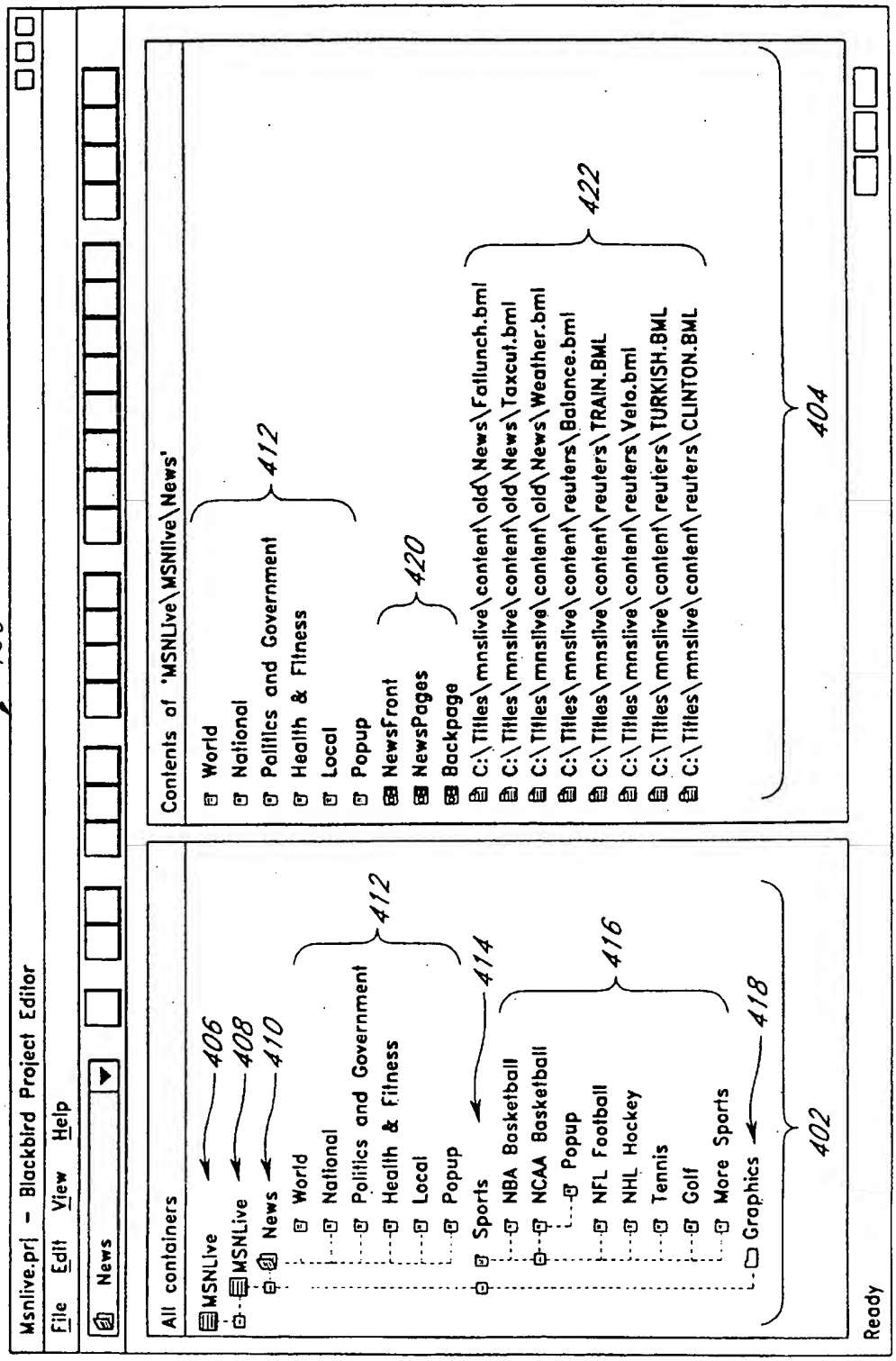
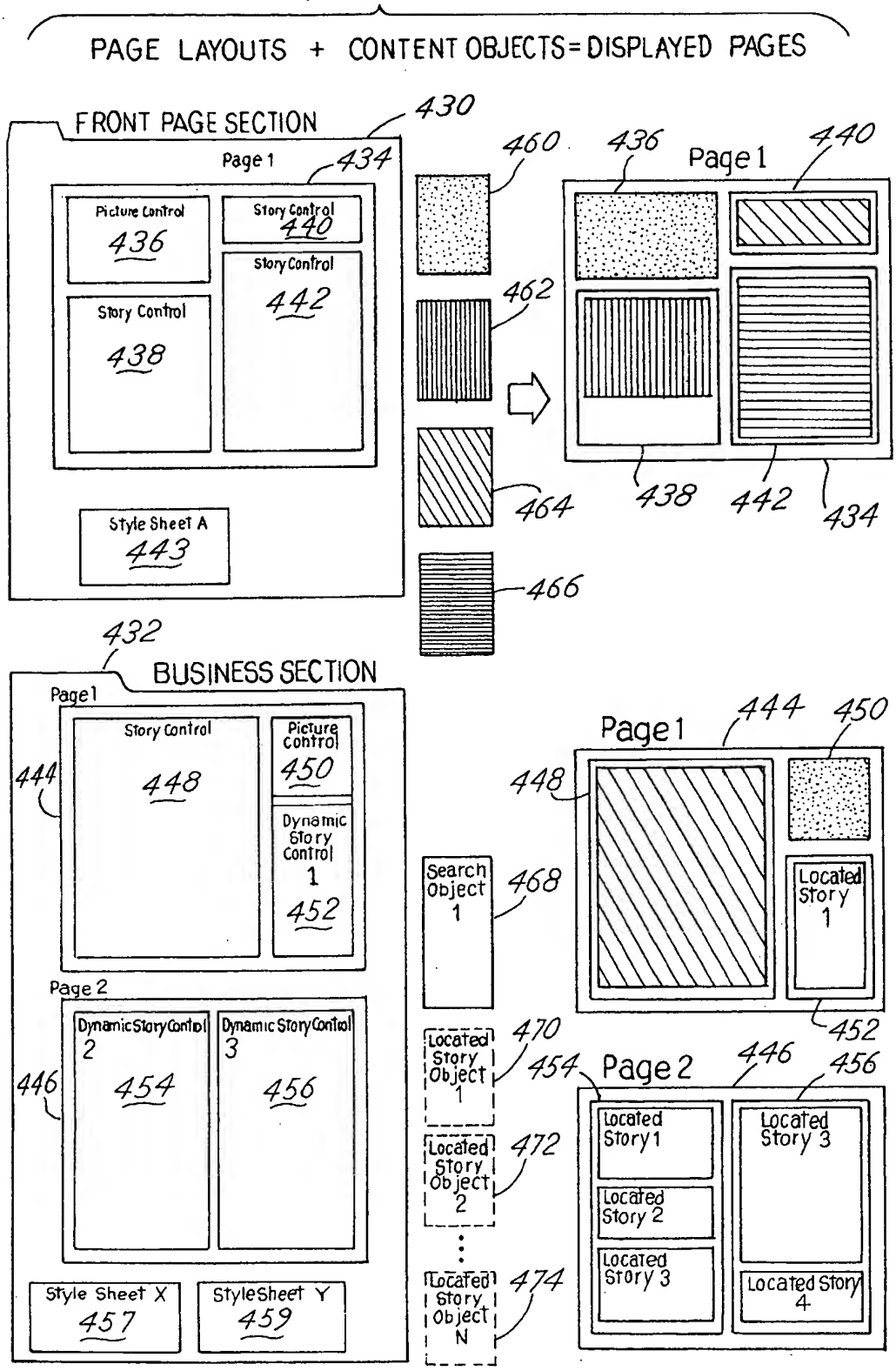

400
Msnlive.prj - Blackbird Project Editor
File Edit View Help
News
All containers
MSNLive
MSNLive
News
World
National
Politics and Government
Health & Fitness
Local
Popup
Sports
NBA Basketball
NCAA Basketball
Popup
NFL Football
NHL Hockey
Tennis
Golf
More Sports
Graphics
406
408
410
412
414
416
418
402
Contents of 'MSNLive\MSNLive\News'
World
National
Politics and Government
Health & Fitness
Local
Popup
NewsFront
NewsPages
Backpage
C:\Titles\mnslive\content\old\News\Fallunch.bml
C:\Titles\mnslive\content\old\News\Taxcut.bml
C:\Titles\mnslive\content\old\News\Weather.bml
C:\Titles\mnslive\content\reuters\Balance.bml
C:\Titles\mnslive\content\reuters\TRAIN.BML
C:\Titles\mnslive\content\reuters\Veto.bml
C:\Titles\mnslive\content\reuters\TURKISH.BML
C:\Titles\mnslive\content\reuters\CLINTON.BML
412
420
422
404
Ready

FIG.7

FIG. 8


PAGE LAYOUTS + CONTENT OBJECTS = DISPLAYED PAGES
FRONT PAGE SECTION
430
Page 1
434
Picture Control
436
Story Control
440
Story Control
442
Story Control
438
Style Sheet A
443
460
462
464
466
436
Page 1
440
438
442
434
432
BUSINESS SECTION
Page 1
Story Control
448
444
Picture Control
450
Dynamic Story Control 1
452
Page 2
Dynamic Story Control 2
454
446
Dynamic Story Control 3
456
Style Sheet X
457
StyleSheet Y
459
Search Object 1
468
Located Story Object 1
470
Located Story Object 2
472
Located Story Object N
474
Page 1
444
450
448
Located Story 1
452
Page 2
454
446
456
Located Story1
Located Story 2
Located Story 3
Located Story 3
Located Story 4

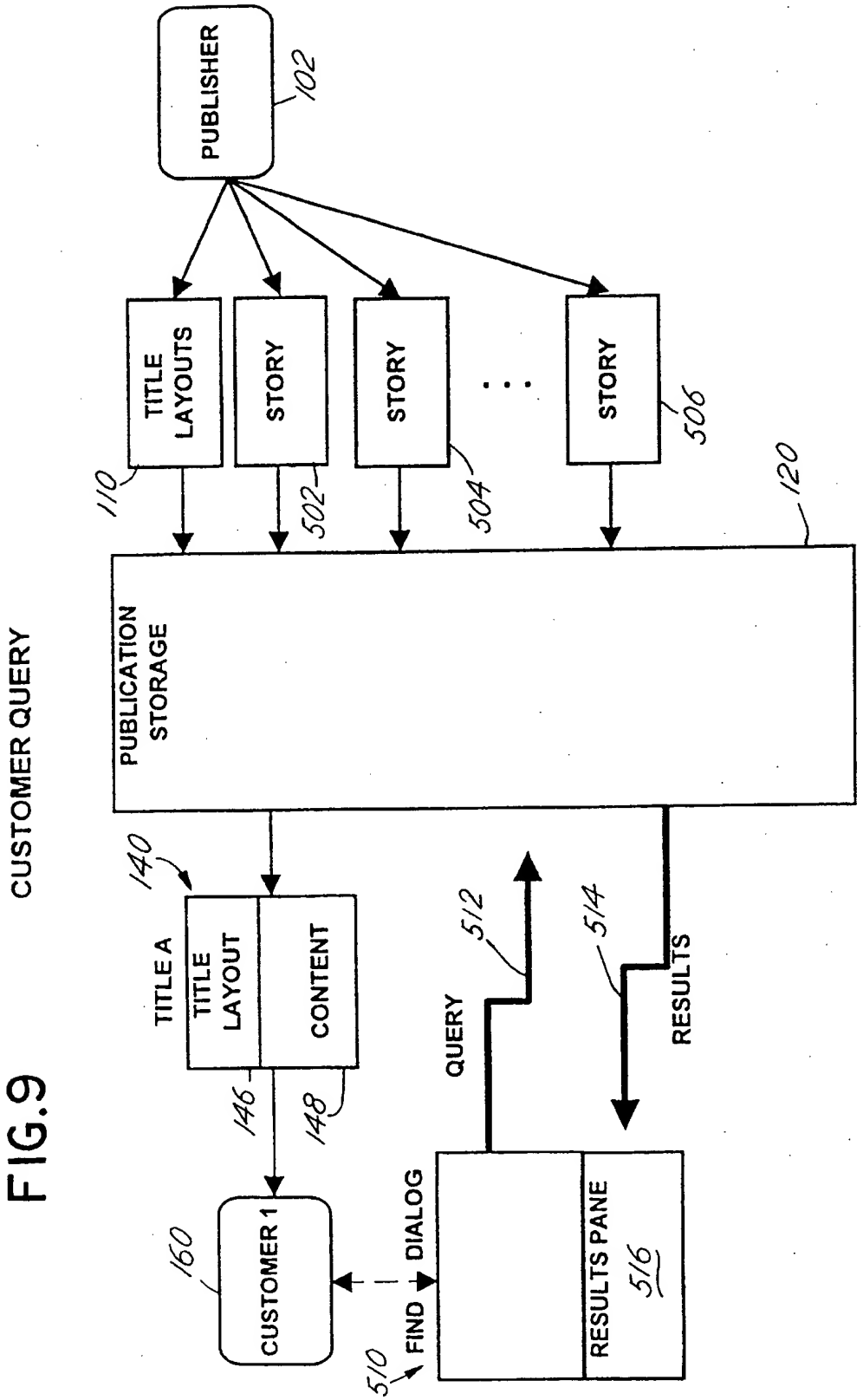

PUBLISHER
102
TITLE LAYOUTS
STORY
STORY
STORY
506
110
502
504
120
PUBLICATION STORAGE
140
TITLE A
TITLE LAYOUT
CONTENT
146
148
CUSTOMER QUERY
QUERY
512
514
RESULTS
160
CUSTOMER 1
DIALOG
FIND
510
RESULTS PANE
516

FIG.9

FIG.10 API AND DLL VIEW OF SYSTEM

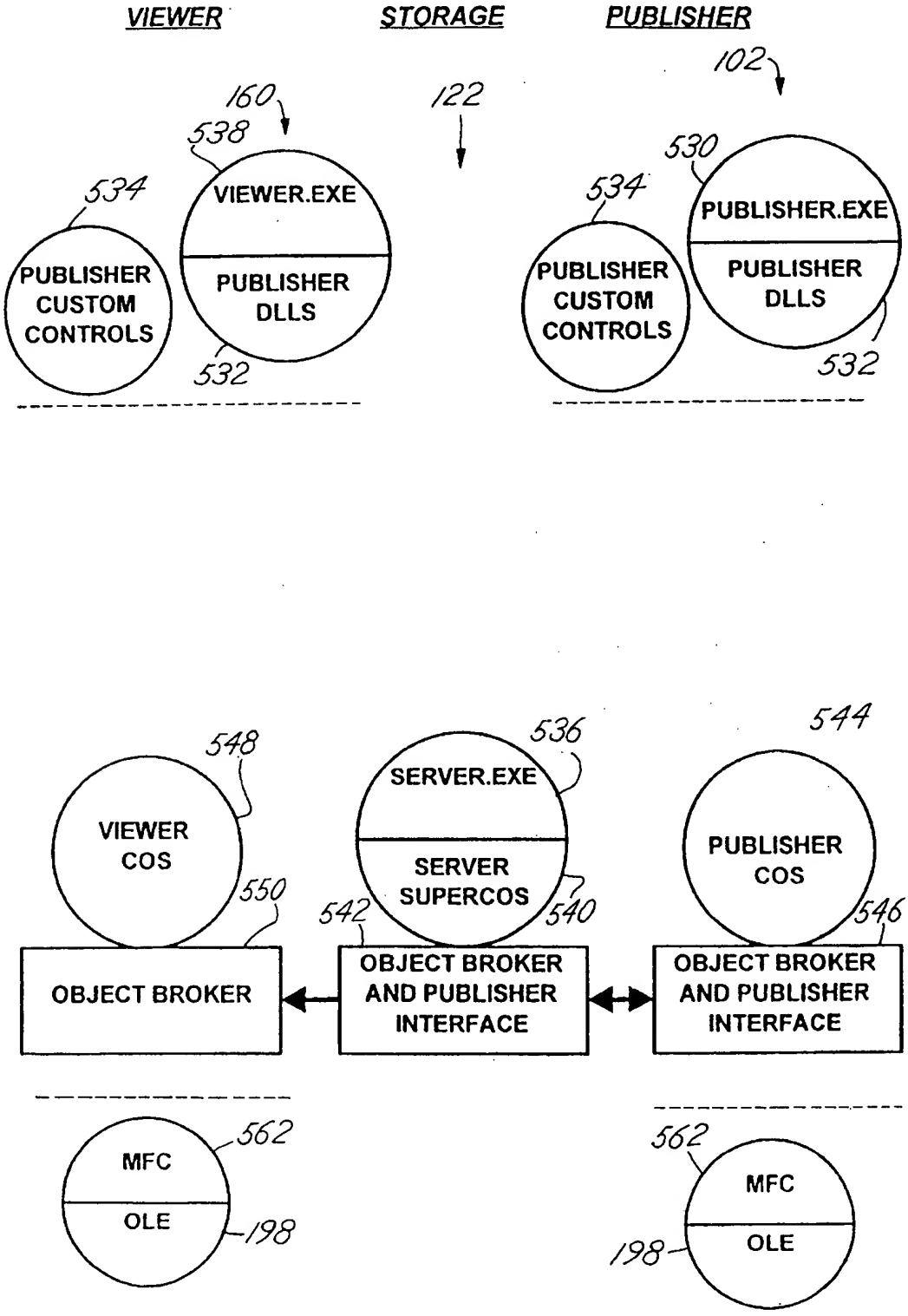
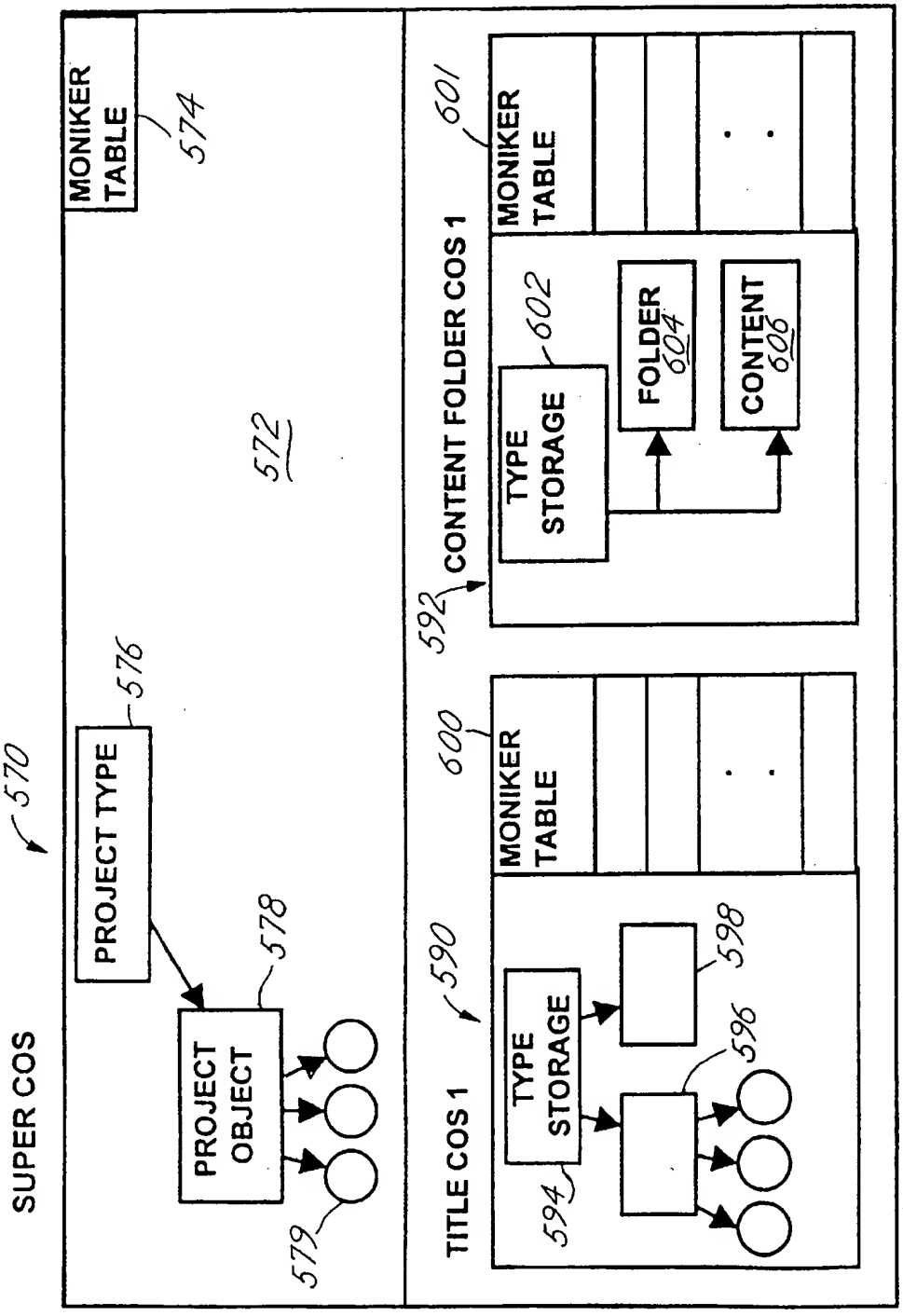


FIG. 11A


570
SUPER COS
MONIKER TABLE
574
572
PROJECT TYPE
576
PROJECT OBJECT
578
579
590
TITLE COS 1
MONIKER TABLE
600
TYPE STORAGE
594
596
598
592
CONTENT FOLDER COS 1
MONIKER TABLE
601
TYPE STORAGE
602
FOLDER 604
CONTENT 606

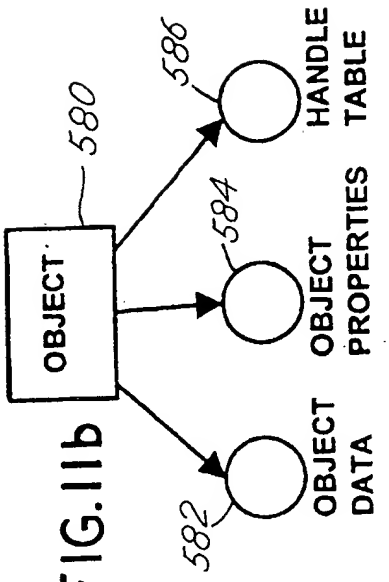


FIG.11b

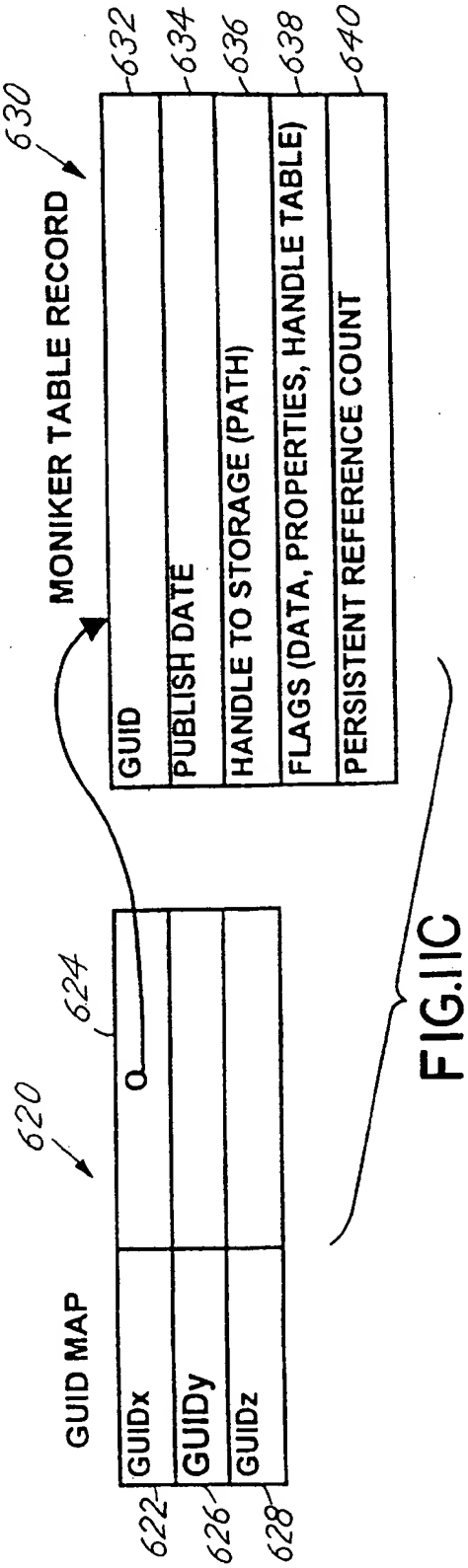


FIG.11C

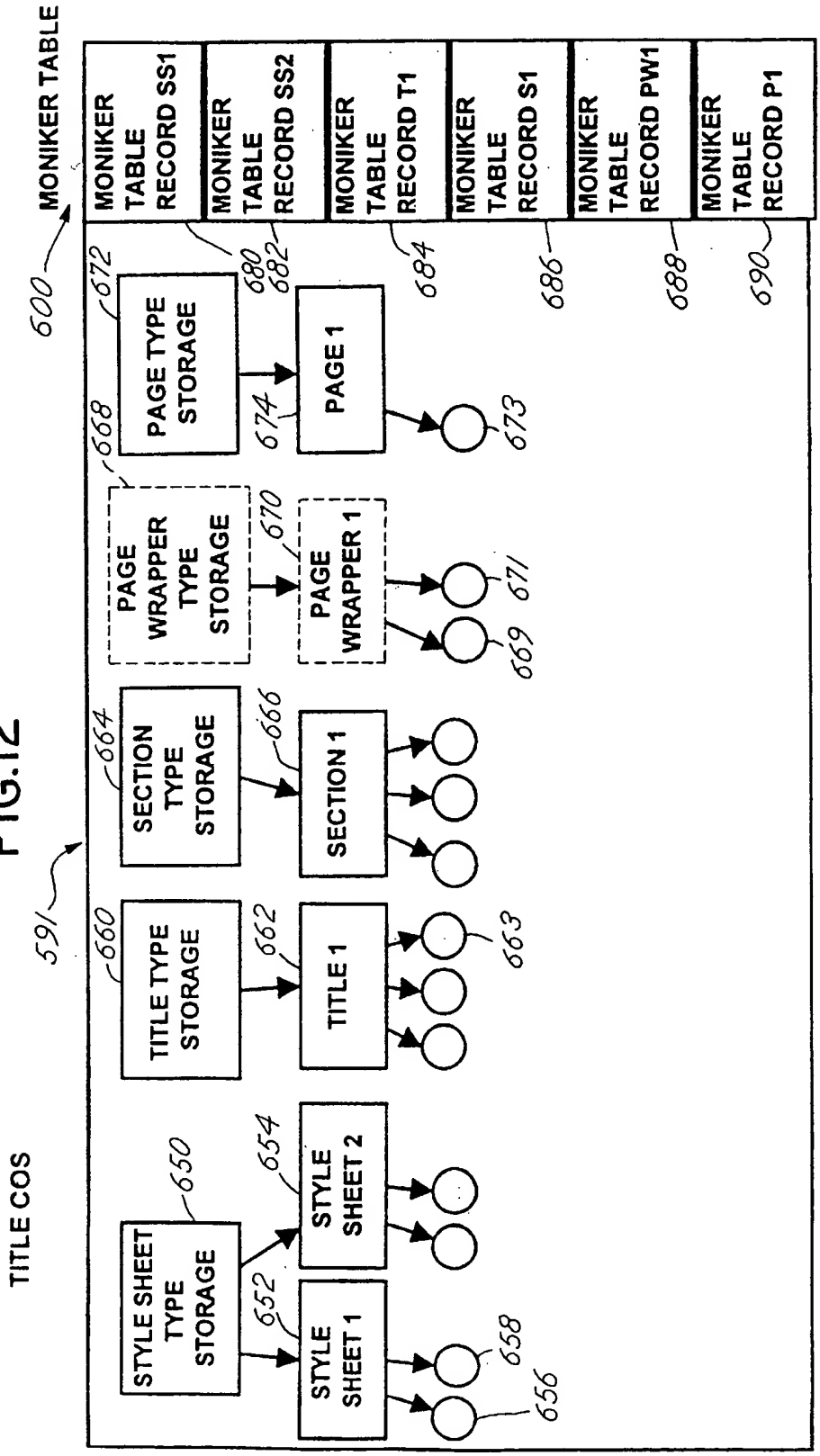

MONIKER TABLE
600
MONIKER TABLE RECORD SS1
MONIKER TABLE RECORD SS2
MONIKER TABLE RECORD T1
MONIKER TABLE RECORD S1
MONIKER TABLE RECORD PW1
MONIKER TABLE RECORD P1
680
682
684
686
688
690
672
PAGE TYPE STORAGE
668
PAGE 1
674
673
PAGE WRAPPER TYPE STORAGE
670
PAGE WRAPPER 1
669
671
SECTION TYPE STORAGE
664
SECTION 1
666
TITLE TYPE STORAGE
660
TITLE 1
662
663
591
TITLE COS
STYLE SHEET TYPE STORAGE
650
STYLE SHEET 2
654
STYLE SHEET 1
652
658
656

FIG.12

TITLE CREATION FLOW DIAGRAM

FIG.13

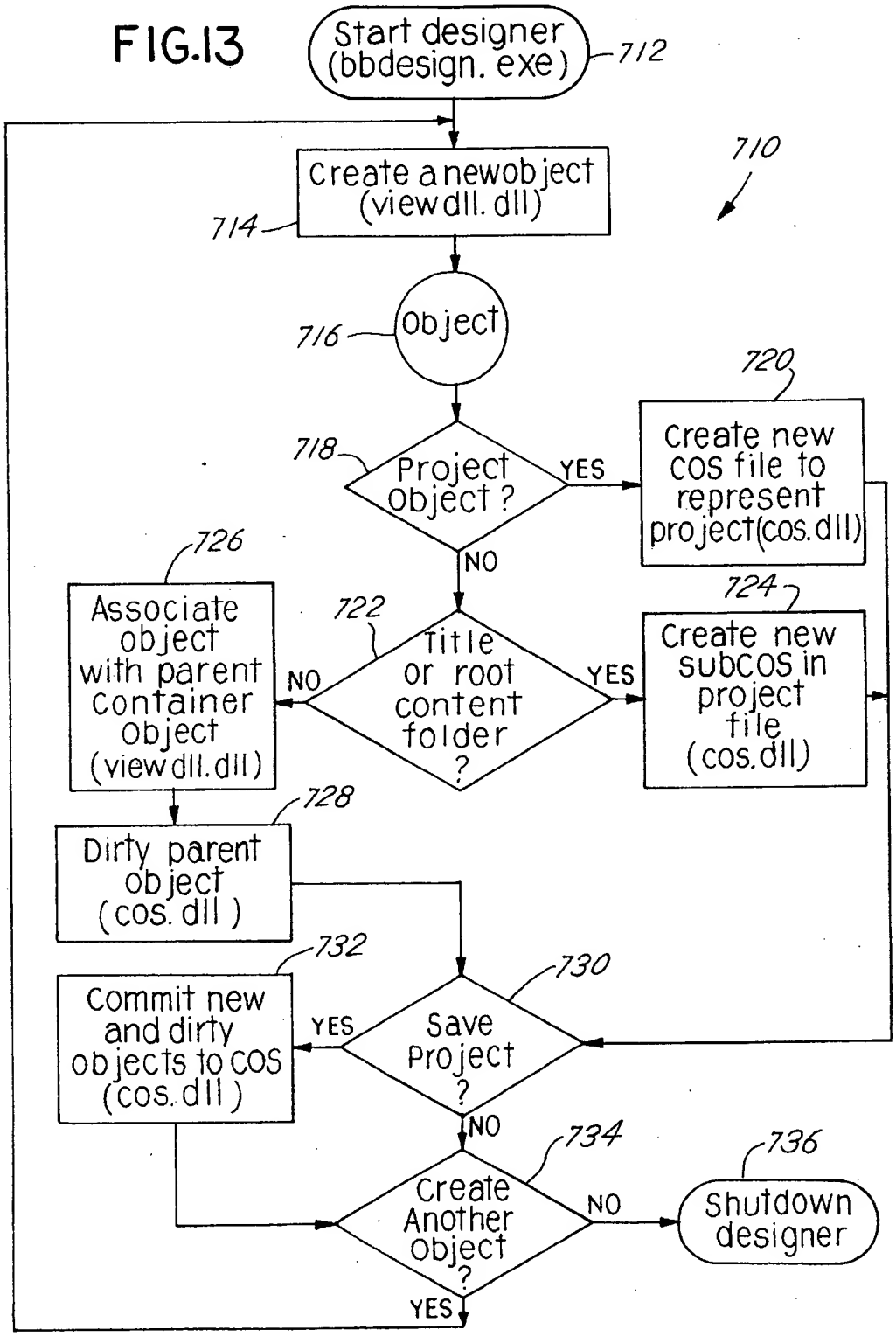
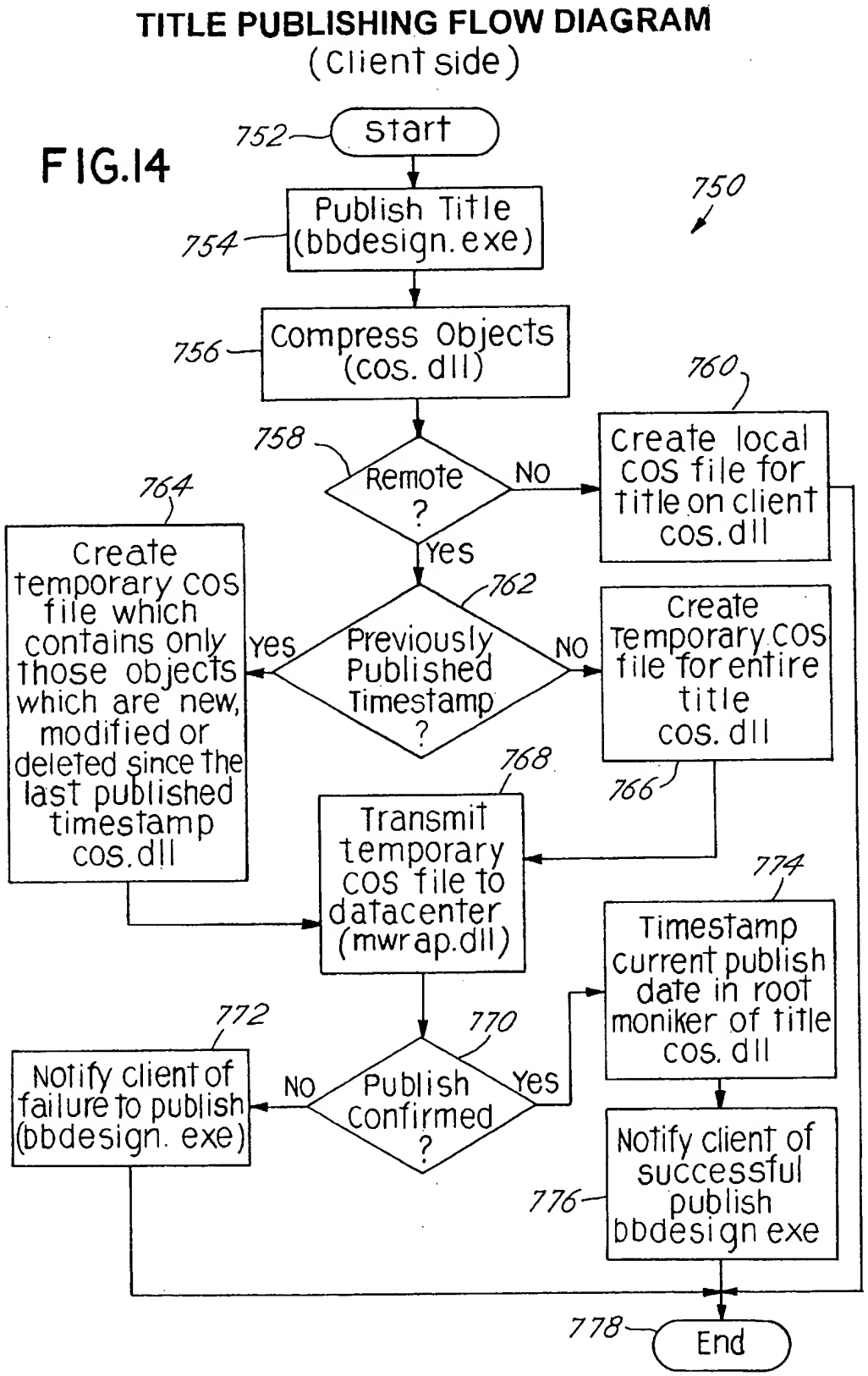


FIG.14


TITLE PUBLISHING FLOW DIAGRAM
(Client side)
750
752
Start
754
Publish Title (bbdesign.exe)
756
Compress Objects (cos.dll)
758
Remote ?
NO
Yes
760
Create local COS file for title on client cos.dll
762
Previously Published Timestamp ?
Yes
NO
764
Create temporary COS file which contains only those objects which are new, modified or deleted since the last published timestamp cos.dll
Create Temporary COS file for entire title cos.dll
766
768
Transmit temporary COS file to datacenter (mwrap.dll)
770
Publish confirmed ?
NO
Yes
772
Notify client of failure to publish (bbdesign.exe)
774
Timestamp current publish date in root moniker of title cos.dll
776
Notify client of successful publish bbdesign exe
778
End

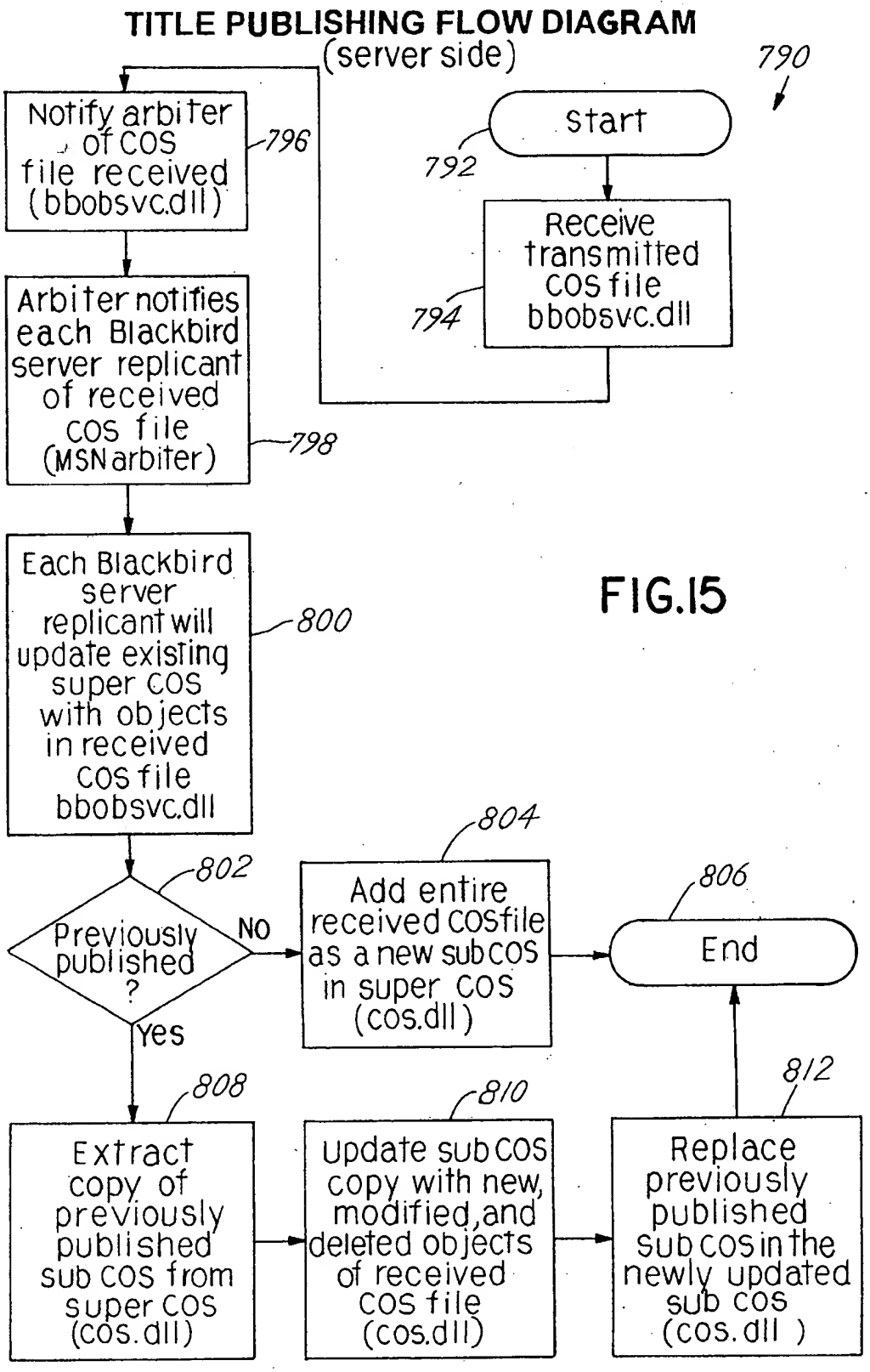

TITLE PUBLISHING FLOW DIAGRAM
(server side)
790
Start
792
Receive transmitted COS file bbobsvc.dll
794
Notify arbiter of COS file received (bbobsvc.dll)
796
Arbiter notifies each Blackbird server replicant of received COS file (MSN arbiter)
798
Each Blackbird server replicant will update existing super COS with objects in received COS file bbobsvc.dll
800
802
Previously published ?
NO
Yes
804
Add entire received COSfile as a new subCOS in super COS (cos.dll)
806
End
808
Extract copy of previously published sub COS from super COS (cos.dll)
810
Update sub COS copy with new, modified, and deleted objects of received COS file (cos.dll)
812
Replace previously published sub COS in the newly updated sub COS (cos.dll)

FIG.15

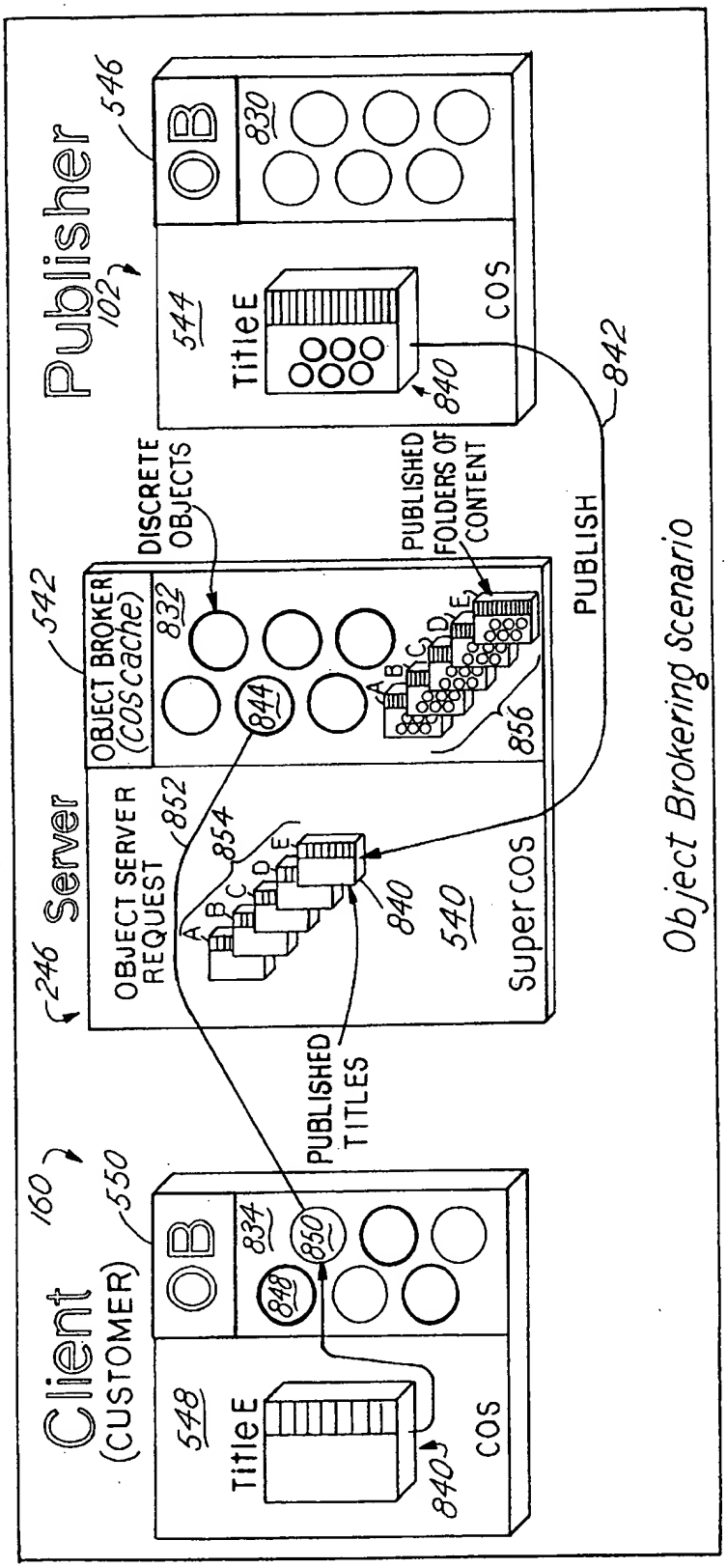
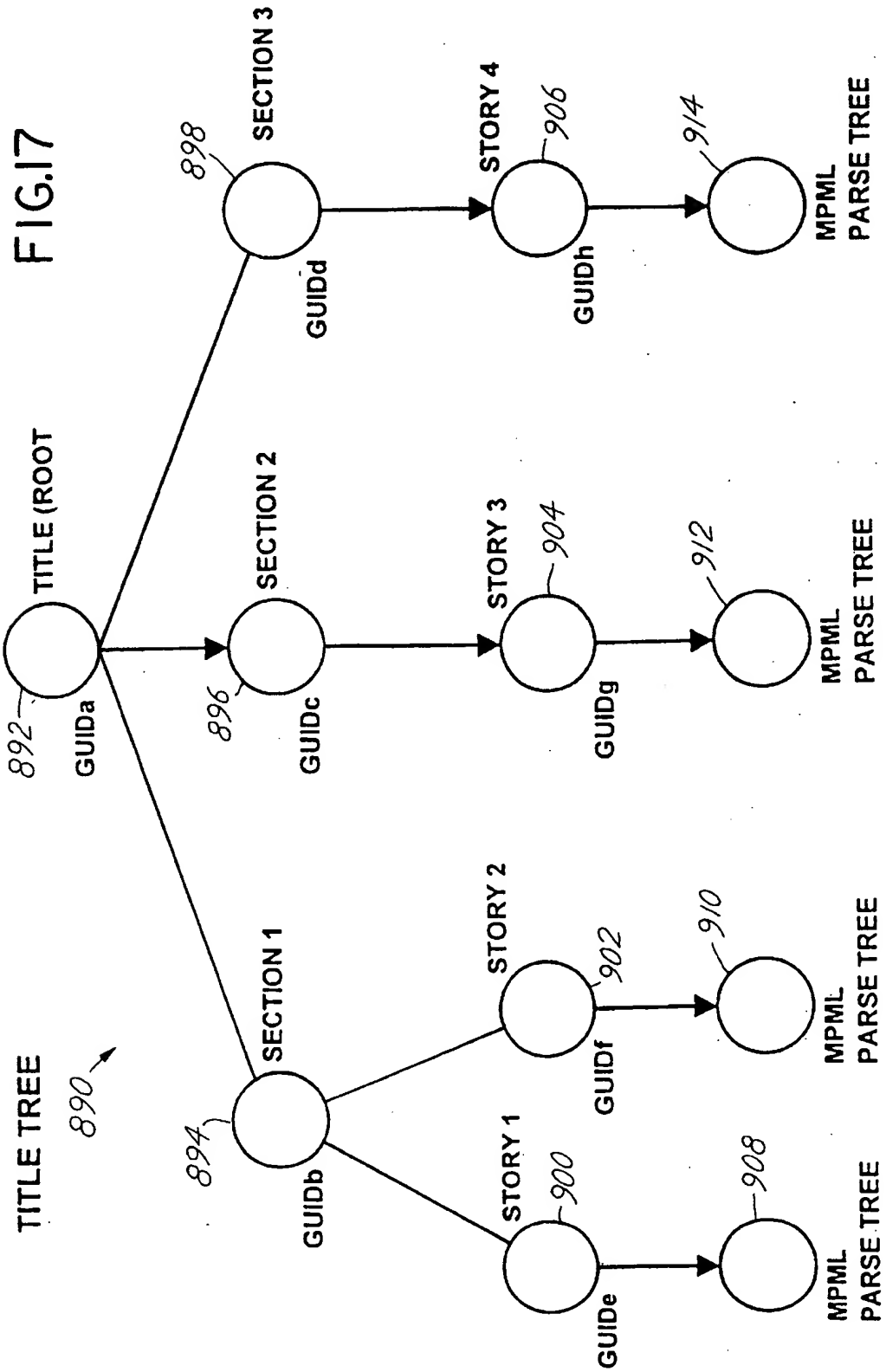

Object Brokering Scenario
Client
(CUSTOMER)
160
550
OB
834
848
850
548
TitleE
840
COS
PUBLISHED TITLES
246
Server
OBJECT SERVER REQUEST
852
854
A
B
C
D
E
840
540
SuperCOS
542
OBJECT BROKER (COS cache)
832
844
DISCRETE OBJECTS
PUBLISHED FOLDERS OF CONTENT
856
PUBLISH
842
Publisher
102
546
OB
830
544
TitleE
840
COS

FIG.16

FIG.17


TITLE TREE
890
TITLE (ROOT
892
GUIDa
SECTION 1
894
GUIDb
SECTION 2
896
GUIDc
SECTION 3
898
GUIDd
STORY 1
900
GUIDe
STORY 2
902
GUIDf
STORY 3
904
GUIDg
STORY 4
906
GUIDh
MPML PARSE TREE
908
MPML PARSE TREE
910
MPML PARSE TREE
912
MPML PARSE TREE
914
MPML PARSE TREE

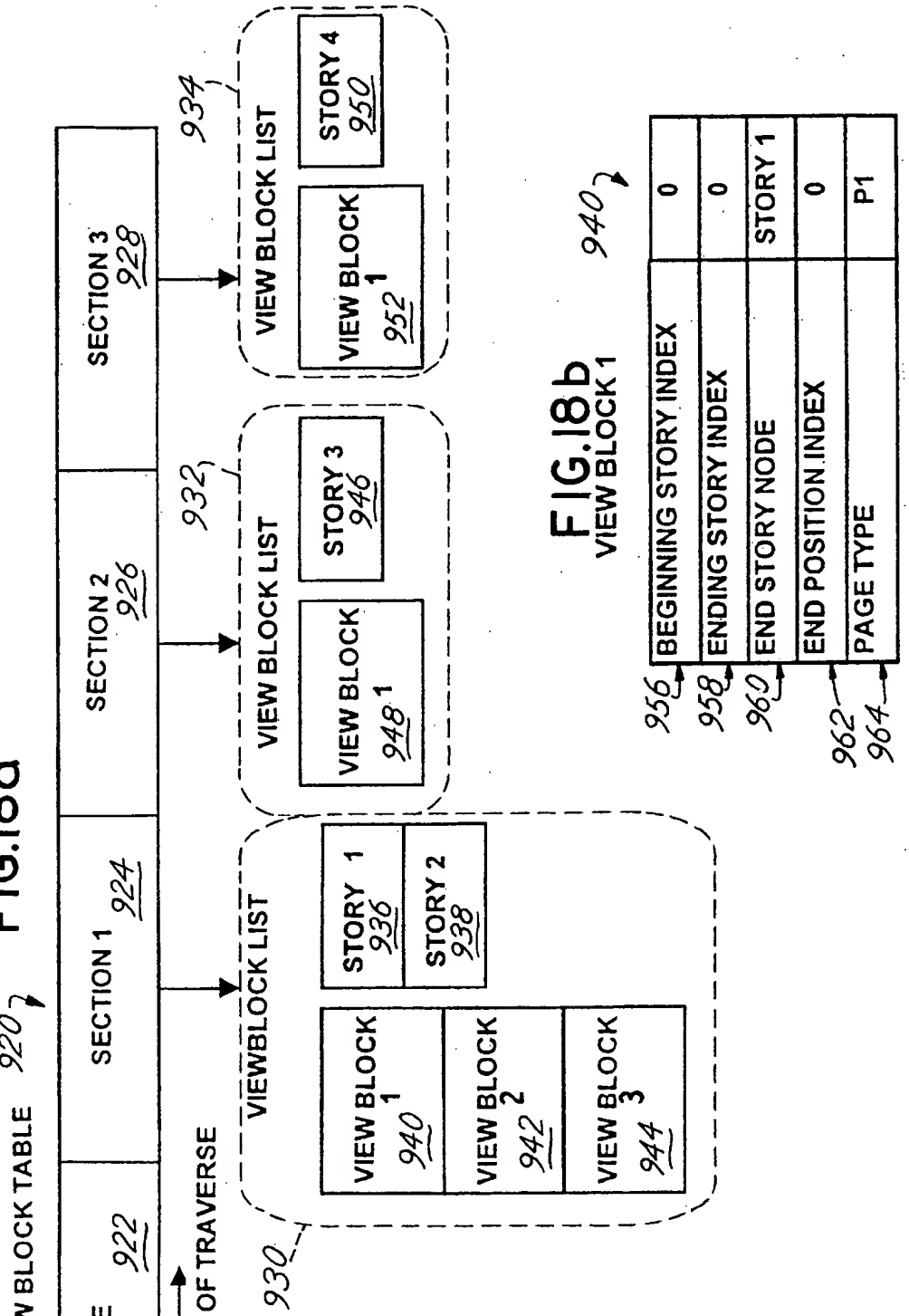


FIG.18a

FIG.18b
VIEW BLOCK 1

FIG.19

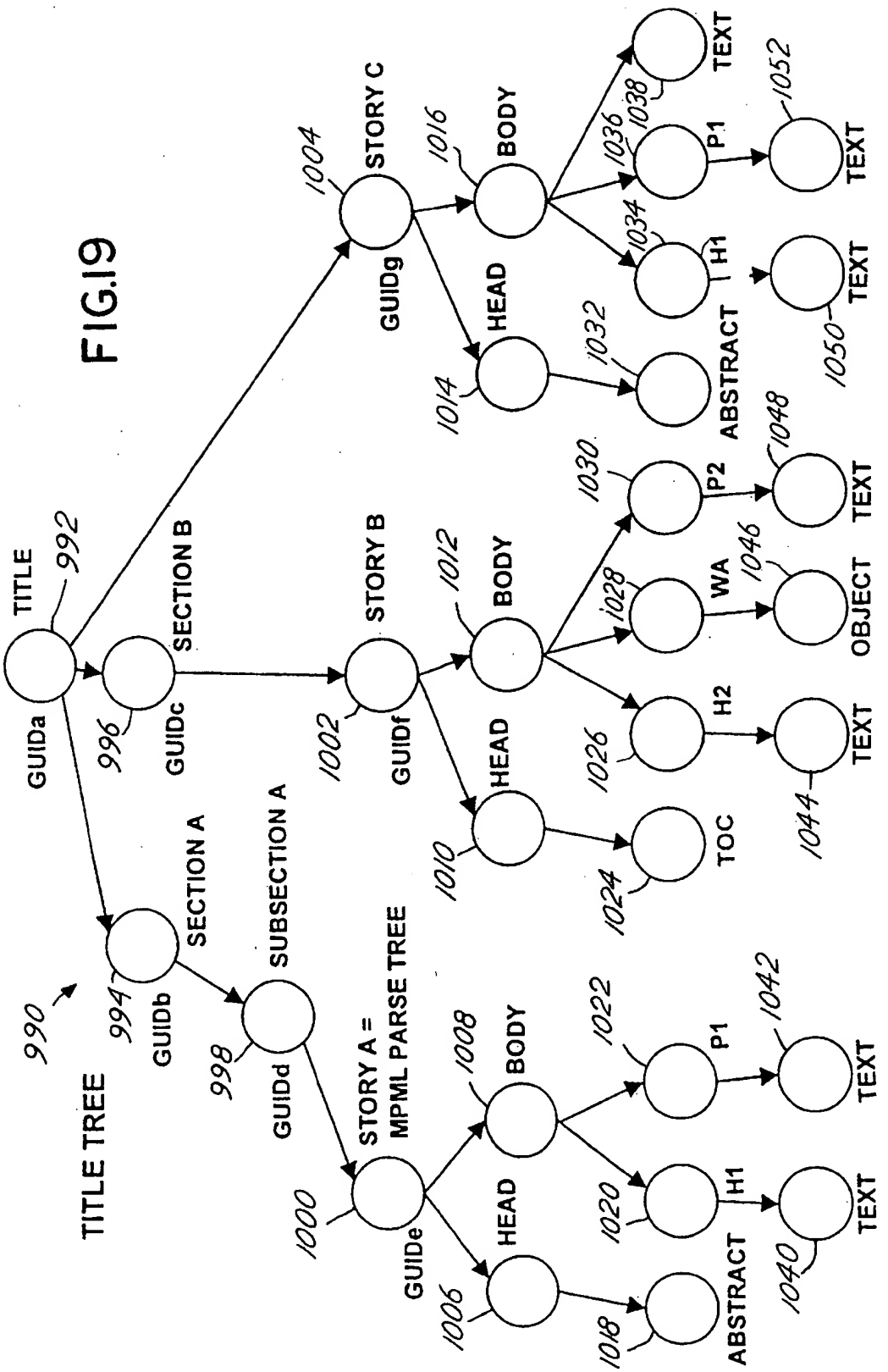

990
TITLE TREE
994
GUIDb
SECTION A
998
GUIDd
SUBSECTION A
1000
GUIDe
STORY A = MPML PARSE TREE
1006
HEAD
1008
BODY
1018
ABSTRACT
1020
H1
1022
P1
1040
TEXT
1042
TEXT
996
GUIDc
SECTION B
1002
GUIDf
STORY B
1010
HEAD
1012
BODY
1024
TOC
1026
H2
1028
WA
1030
P2
1044
TEXT
1046
OBJECT
1048
TEXT
992
TITLE
GUIDa
1004
GUIDg
STORY C
1014
HEAD
1016
BODY
1032
ABSTRACT
1034
H1
1036
P1
1038
TEXT
1050
TEXT
1052
TEXT

FIG.20a

TITLE VIEWING FLOW DIAGRAM

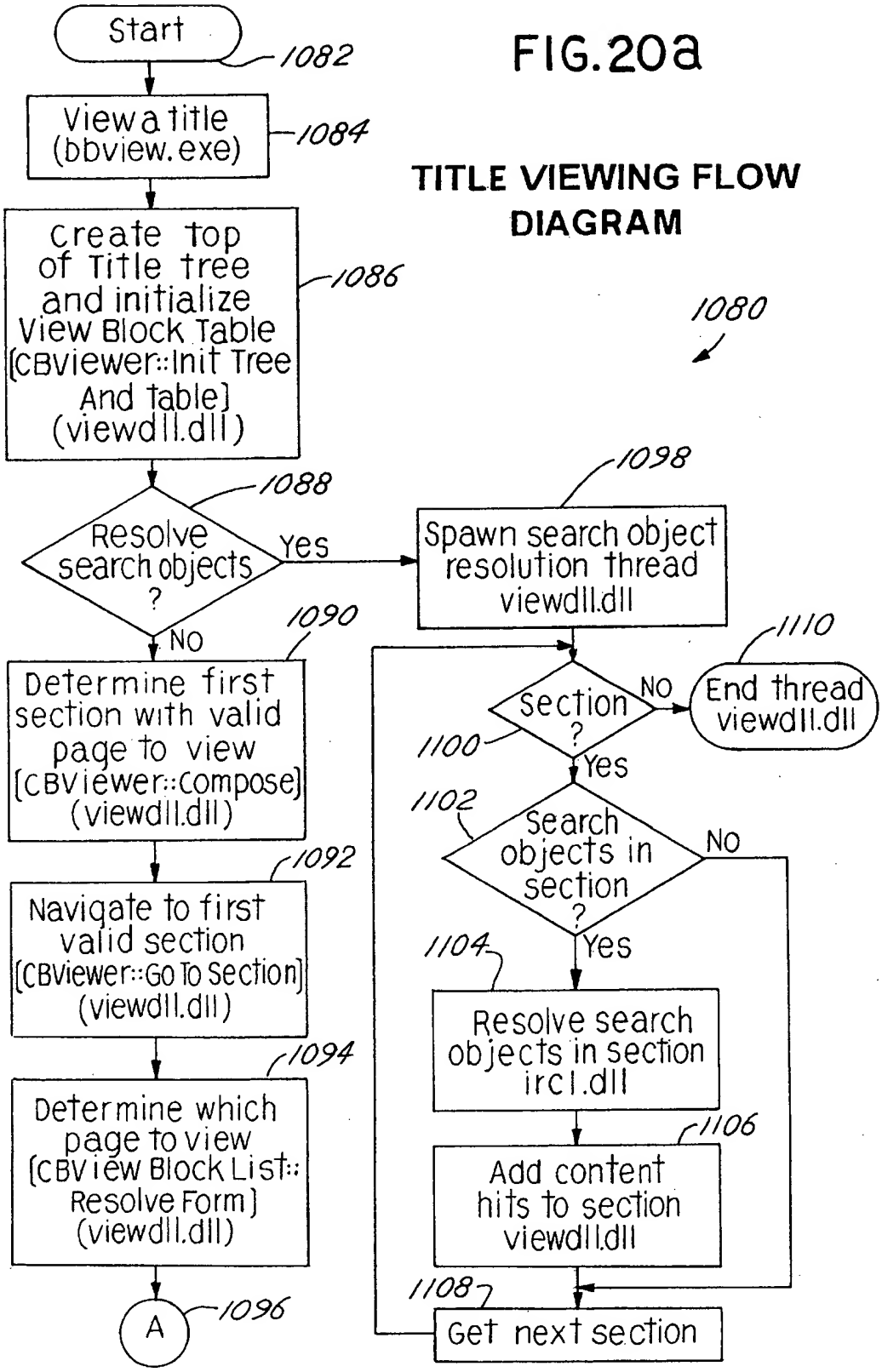


FIG.20b

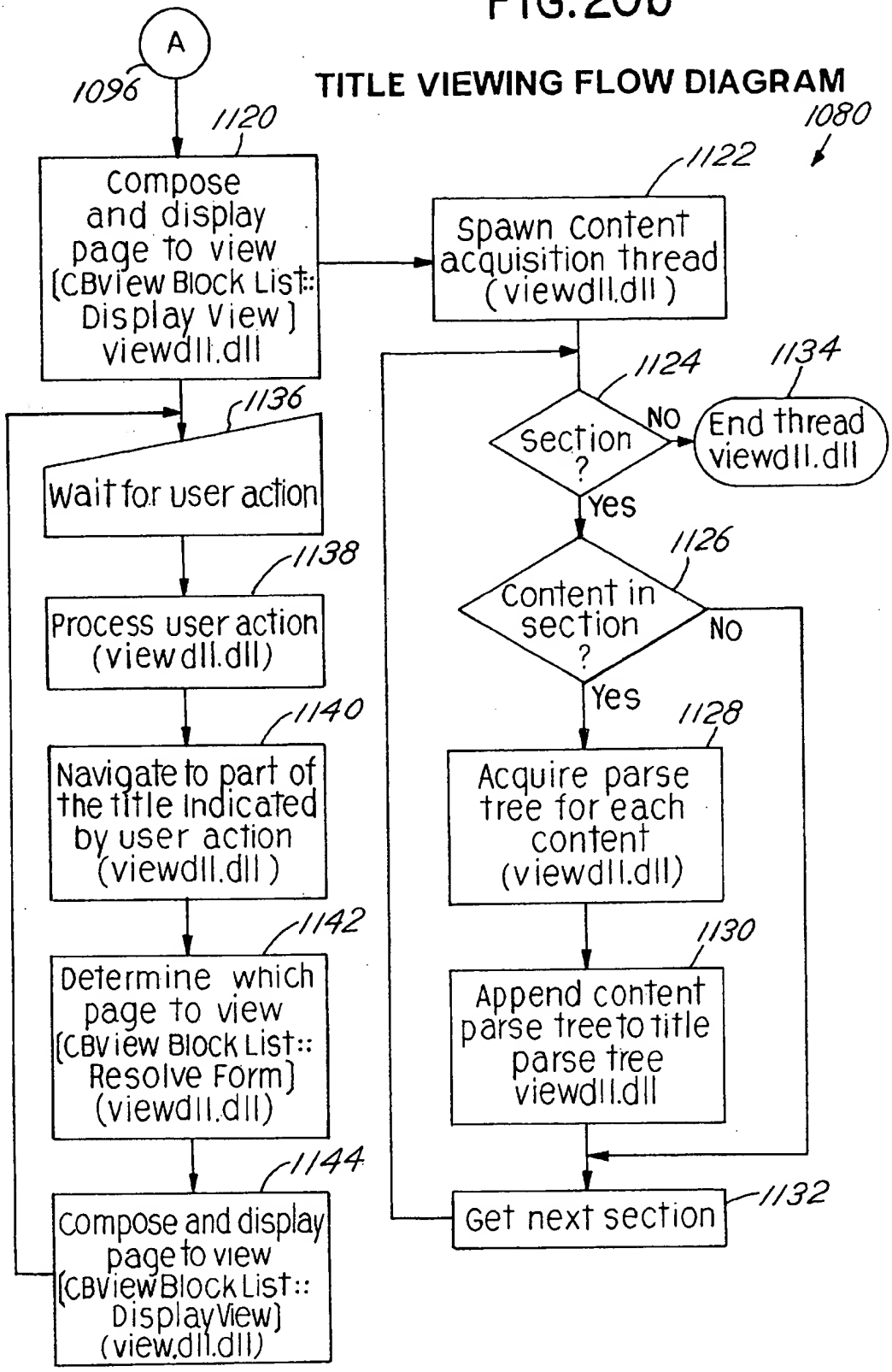
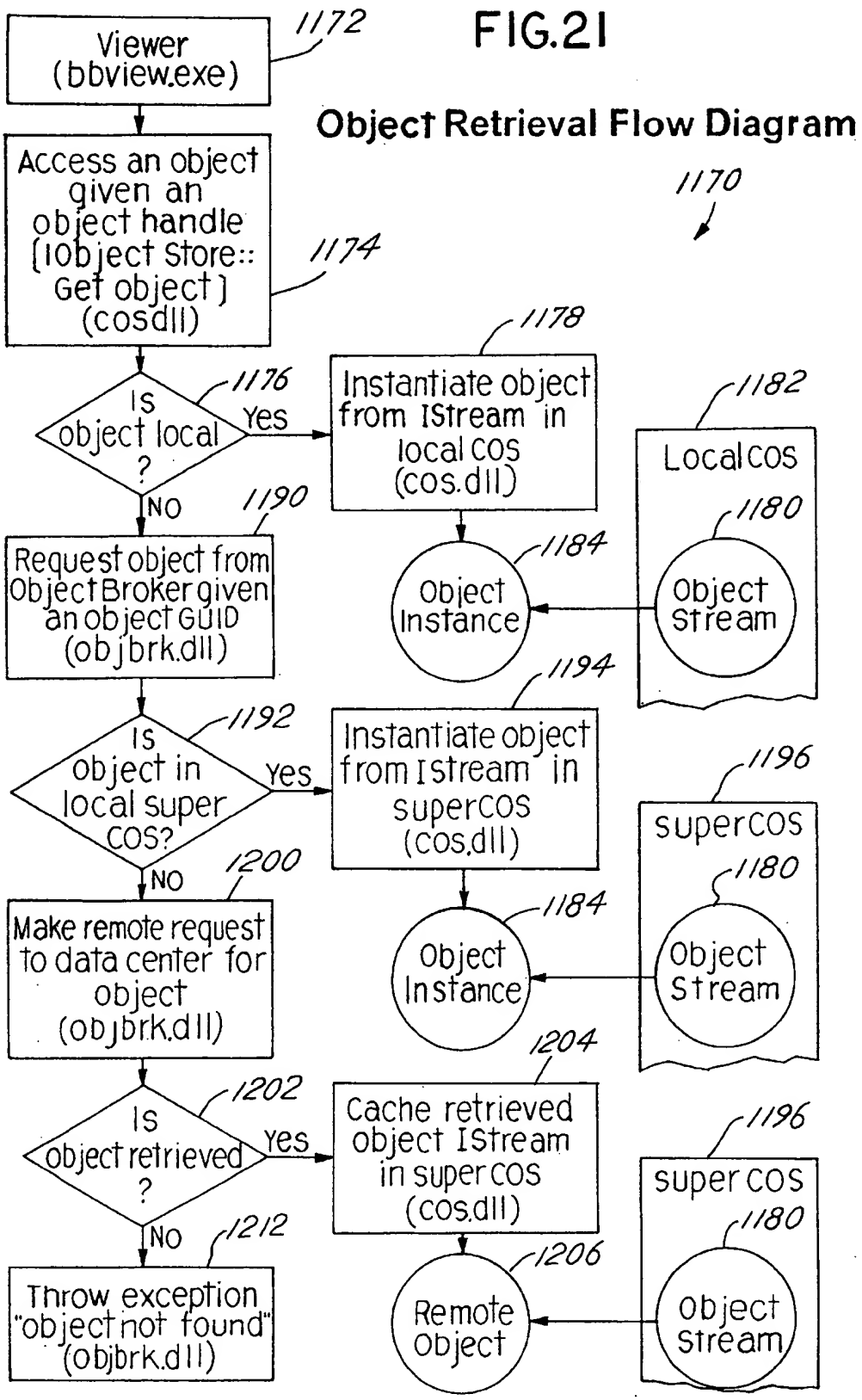

TITLE VIEWING FLOW DIAGRAM
1080
A
1096
1120
Compose and display page to view (CBView Block List:: Display View) viewdll.dll
1122
Spawn content acquisition thread (viewdll.dll)
1124
Section ?
NO
1134
End thread viewdll.dll
Yes
1126
Content in section ?
NO
Yes
1128
Acquire parse tree for each content (viewdll.dll)
1130
Append content parse tree to title parse tree viewdll.dll
1132
Get next section
1136
Wait for user action
1138
Process user action (viewdll.dll)
1140
Navigate to part of the title indicated by user action (viewdll.dll)
1142
Determine which page to view (CBView Block List:: Resolve Form) (viewdll.dll)
1144
Compose and display page to view (CBView Block List:: Display View) (view.dll.dll)

FIG.21


Object Retrieval Flow Diagram
1170
Viewer (bbview.exe)
1172
Access an object given an object handle [IObject Store:: Get object] (cosdll)
1174
Is object local ?
1176
Yes
NO
Instantiate object from IStream in local COS (cos.dll)
1178
Local COS
1182
1180
Object Instance
1184
Object Stream
Request object from Object Broker given an object GUID (objbrk.dll)
1190
Is object in local super COS?
1192
Yes
NO
Instantiate object from IStream in superCOS (cos.dll)
1194
superCOS
1196
1180
Object Instance
1184
Object Stream
Make remote request to data center for object (objbrk.dll)
1200
Is object retrieved ?
1202
Yes
NO
Cache retrieved object IStream in superCOS (cos.dll)
1204
super COS
1196
1180
Remote Object
1206
Object Stream
Throw exception "object not found" (objbrk.dll)
1212

METHOD FOR DELIVERING SEPARATE DESIGN AND CONTENT IN A MULTIMEDIA PUBLISHING SYSTEM

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to multimedia publishing systems and more particularly, to a system and method for publishing and viewing titles which include separate content and layouts.

2. Description of the Related Technology

Microsoft Network, Internet, Compuserve, Prodigy, and America On-line are examples of on-line networks. End users typically access these networks using a microcomputer equipped with a modem. During an on-line session, a user can use a variety of information-related services and communications services, including news services, weather services, bulletin board services, E-mail, and the like.

While on-line services are becoming increasingly popular, today's on-line applications are still in their infancy. In fact, significant problems continue to block independent content providers or publishers from deploying the type of sophisticated and compelling services that are necessary to provide a sustainable on-line business. At the same time, providers of existing on-line services are working to find the right technical business model and usability solutions that will promote acceptance beyond just an early-adopter audience.

In any large city, it is impossible for a single individual to keep up with the activities and events unfolding in the community. Consequently, people turn to writers, reporters, editors, critics, and others, for help in understanding and structuring the information available. In a related trend, broadcast media are increasingly unable to satisfy the needs of a diverse populace. Consequently, in most markets, narrowcast media (media that have tailored and distributed their content to smaller, well defined audiences) have become increasingly popular and profitable. In the on-line community this trend will be correspondingly more important.

One problem content providers encounter when creating applications for the mass market is the diverse audience. For example, some customers will be interested in games, some in, business, some in computer technology, and some in movies. What information should content providers deliver to keep their customers satisfied? What is needed is a system that enables a content provider to create applications that blend the content provider's editorial voice with individual customization. For example, from within a particular application, a customer could indicate an interest in the is computer business and/or classical music, and be able to acquire additional information focused on these areas. Similarly, an on-line publication might automatically synthesize and prioritize content based on different consumer preferences.

There are several significant hurdles facing the on-line industry. These problems include:

Quality. Today's on-line offerings lack the sophistication required to attract and maintain a loyal customer following. Today, the technology simply does not exist to create truly compelling applications and services with rich, interactive multimedia features, while at the same time delivering these applications and services over low-bandwidth connections. What is needed is a system that overcomes these limitations, removing existing design constraints and allowing content providers to easily deploy and maintain a new generation of on-line multimedia applications.

Control. Existing on-line services do not provide content providers complete control over the creation of their on-line applications and services. For example, application creation, distribution, customer relationships, advertising and pricing are most often controlled not by the content provider, but rather by the on-line service itself. In addition, no existing on-line service or the Internet's World-Wide-Web allow content providers to create unique, compelling and highly branded applications. What is needed is a system that enables content providers to create and develop their own unique brands, customer and advertiser relations, and business models. Content providers could then make their branded products the focus of customer interactions to the point that customers often may not be aware they are using an on-line service.

Cost. Providers of on-line services are finding that the tools to effectively manage the process of creating, deploying, and maintaining on-line applications and services are limited. Because existing tools do not fully meet the new demands of the on-line world, the ongoing cost of maintaining on-line applications can be prohibitive. What is needed is a system that is specifically designed to support existing business processes and industry standard information interchange formats. Content providers could then create on-line multimedia titles rapidly and with little production overhead.

Additional concerns for an on-line service include flexibility, automatic delivery of applications to the customer, integrating on-line information browsing with agent-based information gathering, customized content, customer receiving-device independence, support for standards, inclusion of advertising, and a familiar production process.

As more content providers move from the realm of print publishing into the on-line world, they are increasingly alarmed by the real constraints placed upon their ability to create an on-line title that is as visually rich and polished as they were able to create on paper. Further, they would like to take advantage of the multimedia capabilities of today's personal computers to enhance their titles well beyond their paper versions. Specific roadblocks they encounter are:

- In the real world, content authors are rarely skilled graphic designers (and vice versa), yet existing multimedia authoring tools require the same person to do both jobs. At the very least, the same tool is generally used for both jobs, creating an opportunity for errors to be introduced.
- Each authored piece of content must go through the layout process before it is published, a time and human-labor-intensive process that constrains a publisher's ability to provide "real-time" information that is also visually compelling. Also, depending on exactly where a new piece of content is to appear in a publication, other parts of the publication that were previously downloaded may no longer be valid and may need to be re-composed and the new version downloaded.
- Graphics and multimedia objects are huge, and take long periods of time to download to the average customer's or consumer's personal computer across a typical modem. Traditional approaches to visually rich on-line publishing require rendering the screen images on the publisher's machine or on the on-line service's mainframe, which then results in the download of fully-rendered graphics, the worst-case scenario for download time. In this context, rendering refers to the

creation of a bitmap of a display screen in memory prior to displaying the screen. In addition, while many titles contain repeated text and/or pictures, traditional on-line publishing methods require downloading the text or image again each time it appears in a new screen.

- The personal computers that consumers are buying today contain sophisticated processors which can do a remarkably good job of rendering rich, visually compelling titles. However, the current approaches to building, delivering and viewing rich, multimedia titles do not utilize the rendering capability of the consumer's machine.

Content providers would like to support different form factors for displaying the same content—for example, a personal digital assistant (PDA) or a 1024×768 pixel screen. Rich, visually compelling titles would have completely different layouts for display on these two different systems.

Content providers might also wish to create both "full" and "lite" versions of their titles, where the "lite" version contains less content for a smaller price.

Additionally, accessibility concerns might require that a content provider create a "large print" title with a completely different layout, or a title with larger controls for persons with less fine motor control of their hands.

Traditionally, supporting different form factors, "lite" versions, and "large print" titles has required creating a separate copy of the content for each title. Doing so adds additional storage overhead to the system, as well as requiring more work to keep the copies identical when updates occur.

Many different systems exist for publishing documents on a computer system. These systems are used to, for example, create newsletters or brochures to promote a particular company. In addition, publications can be used to disseminate information to a variety of customers. A number of programs exist for allowing a user to design complicated layouts for a particular application. Well-known programs such as Microsoft Publisher®, Ventura Publisher®, PageMaker®, and PrintShop® help a user to produce attractive newsletters and brochures.

These publication systems let the user define particular regions of every page for a specific purpose. For example, the user can place a graphic frame that runs along the top of the page to hold a particular image. Such an image may include the title of the newsletter or another related aspect of the newsletter. In a similar way, the user may define other areas of the first page to include one or more text frames for holding text-based information such as the words from particular story. The user designs the text frame to have certain properties, such as height, width, background color, foreground color and other such properties so that the text becomes attractively formatted for the customer. In addition, the user can format the text information within the text frame to have desired font and paragraph characteristics. For example, the user can highlight the characters within the text frame and define that font to be, for example, bold-faced. The user can also choose to only apply a character format to specific words or paragraphs within a text frame.

After defining an initial text frame in these publishing systems, the user can define additional text frames on the same page. For example, one text frame may hold the title of a story whereas the next text frame holds the name of the author and the text of the story. Although this layout is straightforward to prepare, it is also very difficult to modify once it has been produced.

Another category of publication systems include software for electronically publishing stories across on-line networks such as CompuServe, America On-Line, or the Internet. Most of these systems create and display stories that are formatted in a Standard Generalized Markup Language (SGML) or Hypertext Markup Language (HTML). Both the HTML and SGML are standards for tagging text in documents to be displayed in an on-line network. Documents that are formatted in HTML or SGML can be viewed by several widely distributed browsers such as Mosaic and NetScape for the Internet. These browser programs read SGML and HTML tagged documents and display them with proper formatting. However, the formatting information is stored with the browser and is not distributed by the publisher.

Several programs exist for producing documents that are tagged in either the SGML and HTML format. Programs such as Interleaf's WorldView 2 allow a user to create an SGML document with, for instance, bold-face text and hyperlinks to other documents. Once a document has been saved in an SGML format, it can be read by either the Mosaic or NetScape browser. Unfortunately, all of the formatting commands for text or graphics in an SGML or HTML document are embedded, within the document. The Mosaic or NetScape browsers do not reformat these tagged documents, but rather only display the commands embedded in the SGML or HTML documents to a user. For this reason, the designers that produce the SGML and HTML documents must add formatting commands to every new document. In addition, there is little flexibility to change the document's formatting once the tagged document has been produced. Therefore, the process of creating documents for display using SGML or HTML is very inefficient for the document designer.

Other commercially available software programs for producing on-line publications are available in the marketplace. One type of electronic publisher that generates its own specific format of text while retaining the specific layout of the document is the Adobe Acrobat™ software package. Acrobat™ reads and stores documents in a specialized format known as the Portable Document Format, (PDF) for use on the Internet. Other electronic publishing programs are produced by Interleaf, Inc. (Waltham, Mass.), Farallon Computing (Alameda, Calif.) and Common Ground Software (Belmont, Calif.).

Another on-line information system is described in U.S. Pat. No. 5,347,632 by Filepp et al. This patent discusses an interactive computer system network which enables a user to display news information and perform transactional services through a personal computer. However, in the Filepp system, the news information is integrated into display regions.

The invention described in U.S. Pat. No. 5,347,632 includes procedures for formulating objects that have been specially structured to include display data, control data and program instructions. Unfortunately, this system does not provide a separation of the content being displayed from the design. Therefore, the same design layout cannot be shared among disparate pieces of content.

The content displayed in this system is therefore difficult to modify because new design layouts must be transmitted to the users across slow communications lines for every piece of information viewed on the computer monitor. If the content of the information was separated from the design layout at publication time, then design layout objects could reside locally on the user's computer and be available whenever required by a specific piece of content. These disadvantages are overcome by the present invention.

SUMMARY OF THE INVENTION

This invention is a multimedia publishing system designed for creating publications that include components

for design, authoring, distribution, viewing, search, personalization, and billing of on-line services. The major benefit of such a system is efficient distribution, content published separately from the layout, separation of responsibilities, hardware independence, automatically placed content and personalized titles.

Efficient distribution is achieved by separating the content and design which enables transmission of high-quality titles over low-speed communications links subject to loss of connectivity. Since the design of many titles remains reasonably static while the content changes on a regular basis, caching the layout designs minimizes the transmission of redundant data and optimized the bandwidth use. Typically, content is transmitted quickly since it consists of tagged components, not the actual pages and controls themselves. This significantly reduces transmission times for downloading a title. Also, when the same content is used more than once in a title, reuse of content is possible, further reducing the amount of information that must be downloaded. Once the content object is downloaded, it can appear instantaneously on as many pages as the publisher desires.

Since the content is published separately, this eliminates the requirement that the content to be laid out by a designer before it can be published. Content can be uploaded to the distribution point and downloaded to consumers' machines as soon as the object is completed. Since the rendering is automatically performed on the viewers' computer based upon the designs in the title's page layouts.

For many content providers, especially those creating on-line publications, the separation of design and content fits the existing production process. Many editors have found the amount of production effort required to put content on-line is inhibiting. With this invention, a design can be created only as needed and the changing content is dynamically flowed into the design cached and located in the viewer's computer.

The separation of responsibilities for layout designers and content authors is enhanced during the publication process because the graphic designers can work on the title and page layouts, while separately, the authors can create the content objects.

The tagged content can be displayed with high quality on a variety of different devices. For example, a content provider can create a title just once, but the title can be viewed on a VGA screen with one column, a printer with many columns, a small screen PDA, an interactive television (ITV) system, a fax machine, or a notebook computer. Device independence can represent a significant cost savings for content providers, and can help to ensure that a content provider's applications are able to effectively reach a large audience.

The tagged components of a structured story (the abstracts, headliner, etc.) can be automatically placed in different parts of a title, making it easier for viewers to read and navigate the information. For example, the headline and abstract of a story might be displayed on the front page, and the headline and body of the story might be displayed on an inside page. This dynamic layout reduces the effort required to publish up-to-date, round-the-clock titles.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is block diagram of the basic system configuration of the multimedia publishing system (MPS), which is presently preferred underlying architecture for the present invention.

FIG. **2** is a diagram of the major system components of the MPS shown in FIG. **1**.

FIG. **3** is a diagram of an exemplary on-line system for publication storage and distribution.

FIG. **4** is block diagram of a hierarchy of containers or folder for a plurality of publishers using the system of FIGS. **1** and **2**.

FIG. **5** is a overview flow diagram of the MPS processes performed using the system of FIGS. **1** and **2**.

FIG. **6** is an exemplary screen display of one page of a title as displayed by the viewer of FIG. **2**.

FIG. **7** is an exemplary screen display of the parts of the content and layout for the title displayed in FIG. **6**.

FIG. **8** is a block diagram of the interaction of page layouts, controls, style sheet and content objects at the viewer of FIG. **2**.

FIG. **9** is a diagram illustrating a query performed by the customer using a "find" dialog on the system shown in FIGS. **1** and **2**.

FIG. **10** is a block diagram of the major software components of the system shown in FIGS. **1** and **2**.

FIG. **11***a* is a diagram of a superCOS component of the system shown in FIGS. **1** and **2**.

FIG. **11***b* is a diagram of an IStorage structure used in the implementation of the superCOS of FIG. **11***a*.

FIG. **11***c* is a diagram of a moniker table record and a GUID map used in the implementation of the superCOS of FIG. **11***a*.

FIG. **12** is a diagram of a title COS used in the implementation of the superCOS of FIG. **11***a*.

FIG. **13** is a flow diagram of a title creation process performed on the system shown in FIGS. **1** and **2**.

FIG. **14** is a flow diagram of a title publishing process performed at the publisher's workstation on the system shown in FIGS. **1** and **2**.

FIG. **15** is a flow diagram of a title publishing process performed at the network on the system shown in FIGS. **2** and **3**.

FIG. **16** is a diagram of an exemplary object brokering scenario of the process used on the network of the system shown in FIGS. **2** and **3**.

FIG. **17** is a diagram of an exemplary title tree generated at the viewer component shown in FIG. **2**.

FIG. **18***a* is a diagram of an exemplary view block table and viewblock lists used by the viewer component shown in FIG. **2**.

FIG. **18***b* is a diagram of an exemplary view block of one viewblock list shown in FIG. **18***a*.

FIG. **19** is a diagram of another exemplary title tree, which includes exemplary parse trees, that is generated at the viewer component shown in FIG. **2**.

FIGS. **20***a* and **20***b* are a flow diagram of a title viewing process performed by the system of FIGS. **1** and **2**.

FIG. **21** is a flow diagram of an object retrieval process performed by the title viewing process defined in FIGS. **20***a* and **20***b*.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Reference is now made to the drawings wherein like numerals refer to like parts throughout. For convenience, the following description will be organized into the following six principle sections: Acronyms, Advantages of the Multimedia Publication System, Multimedia Publishing System Overview, Designer Environment at the Publisher, Viewer at the Customer, and Conclusion.

I. ACRONYMS

The following list of acronyms is provided as a reference in reading the remaining sections.

AVI -	Advanced Video Imaging.
BBS -	Bulletin Board System.
MPML -	Multimedia Publishing Markup Language
CF -	Component Forms
COS -	Caching Object Store
DBM -	Database Management System
DLL -	Dynamic-link Library
GUID -	Globally Unique Identifier
HTML -	HyperText Markup Language
ICP -	Independent Content Provider
IM -	Information Magnet
IP -	Information Provider
LAN -	Local Area Network
MP -	Multimedia Publishing
MPC -	Microsoft Network Procedure Call
MPS -	Multimedia Publishing System
MFC -	Microsoft Foundation Class
MSN -	Microsoft Network
OCX -	OLE Control
OFS -	Object File System
OLE -	Object Linking and Embedding
PDA -	Personal Digital Assistant
RPC -	Remote Procedure Call
RTF -	Rich Text Format
SGML -	Standard Generalized Markup Language
VBA -	Visual Basic for Applications
WAN -	Wide Area Network
WWW -	World-Wide Web

II. ADVANTAGES OF THE MULTIMEDIA PUBLICATION SYSTEM

The present invention can perhaps provide the most benefit by using an on-line network. Therefore, this and the following sections present background information on a preferred on-line publication system which is a foundation upon which the present invention can reside.

To enable a new generation of on-line, multimedia applications, an end-to-end system has been invented for developing and using applications and services. The system, called the Multimedia Publishing System (MPS or MP system), preferably uses the Microsoft Network. As an open, turnkey system, the MPS includes components for design, authoring, distribution, viewing, search, personalization, and billing of on-line services and multimedia applications. The MP system allows content providers to offer rich, interactive multimedia applications and services, providing users a compelling and exciting on-line experience. The MP system provides the key to overcoming the previously described hurdles facing the on-line industry.

The Microsoft Network removes the primary barriers to on-line service use. These barriers include cost, difficult user interfaces and lack of inertia. Access to The Microsoft Network is provided by Windows 95, the most recent version of the Microsoft Windows operating system thereby making it accessible to millions of customers. The Microsoft Network is designed to make accessing electronic information easy and inexpensive for any user of Windows 95.

In the MP system, Independent Content Providers (ICPs), also known as publishers, supply the system with stories, publications, newspapers, sounds, graphics movies and much more. The MP system is designed to take projects (e.g. stories, publications, and so forth) produced by the publishers and make them accessible to millions of users on the Microsoft Network. Thus, the basic components of the MP system are a project designer component, a public distribution site, and a viewer component. These components of the MP system are described in detail below.

One unique concept that permeates the MP system is the clean separation of content and design. In this context, content is defined as the actual data that is to be displayed to the user. The design of a project is how that information gets displayed to the user (e.g., its format on the computer screen). An illustrative example would be an electronic newspaper, wherein the content is the text and graphics of the stories, while the design is the layout and style of that data. The design of the electronic newspaper is what makes it look like a newspaper on a computer monitor, whereas the content is the data that makes up the designed screens.

In the MP system, the content and the design are stored as separate objects in the public distribution site so that many different pieces of content can be viewed with the same appearance. An object can be defined as a discrete data item or data structure which can be stored in persistent storage or in memory. The object may include computer instructions for manipulating the data. Once a designer using the project designer component at the publisher site has created a particular page layout that is attractive, many pieces of content can be viewed from within that layout because of the separation of content from design in the MP system. The system keeps track of links between a piece of content and its associated page layout, but does not actually format the data in the content with a particular style.

As will be discussed in more detail below, the designer creates projects with design and content information for a particular publisher. Continuing the example from above, a project could correspond to an entity that owned a series of newspapers and other media businesses. Within each project, one or more titles would correspond to the actual newspaper. Each title has one or more sections, and can be thought of as similar to the sections in a standard, printed daily newspaper or other periodical such as a magazine.

Within each section are pages that define the information that is displayed to a single screen on the customer's computer visual display. When viewing a particular title, the customer will normally look at only one page of information at a time. On each page are controls which contain instructions for gathering, formatting and displaying the linked content onto the page. When a customer looks at information on a page that is provided by a publisher, the customer is really looking at content that has been formatted within pre-defined control regions on the page.

One important facet of this invention is the concept of viewing the same content objects in many different ways. As discussed above, content objects are viewed after being formatted by a particular linked control. The control knows how to format a particular piece of content by looking at the style that has been defined for that content by the designer and then comparing that style to a linked style sheet. Because each control on a page can have a different associated style sheet, different controls on the same page can each display the same linked content in varying formats. In one control, the title might be displayed using a 14 point font and bold emphasis, whereas the same piece of content in a different control on the page can be displayed in a 12 point font and italic emphasis. The ability of each control on a page to have its own associated style sheet is a powerful tool for the designer to use to format attractive content on a page.

Unlike prior publishing systems, content (such as text or graphics) in the MP system is never reformatted into the marked style. The content is only displayed to the user in the

chosen style. Therefore, should the designer choose to change a particular style, only the style sheet property of that style needs to be altered. The next time that the content is displayed using the altered style sheet, the content will be displayed with the properties of the new style. Other advantages and benefits of the MP system are discussed in detail below.

To provide more detail on the advantages of the MP system, the following section presents an overview of the Multimedia Publishing system.

III. MULTIMEDIA PUBLISHING SYSTEM OVERVIEW

This section presents an overview of the configuration and major components of the preferred Multimedia Publication System. Beginning with a description of the important concept of separating design (or title layout) and content, this section continues by discussing the major components and configuration of the MP system. In addition, a description of the container hierarchy is discussed in conjunction with FIGS. **1–4**.

The objects utilized by the MP System include a project; title; content folder and, optionally, subfolder; section and, optionally, subsection; window; page; control; style sheet; and various content objects (such as stories, images, audio, so forth). These objects will be explained in more detail below in reference to FIGS. **1–7**. It is important to realize that these objects need to be stored in a non-volatile computer memory such as a hard disk drive.

The natural way of storing related and ordered objects is in a data structure, such as an acyclic graph. The presently preferred way of storing the MP system objects is called a caching object store (COS). In the presently preferred MPS, each title corresponds to a COS. There is least one COS at the publisher workstation and in each MPS server at the publication storage and distribution center (FIG. **2**). Each customer workstation also has a COS so that the customer can store and retrieve MP system objects when assembling content into controls on pages.

A title may be broadly defined to encompass a publication (e.g., newspaper), service (e.g., stock quotations) or application (e.g., multimedia encyclopedia). When a title is viewed, the viewer opens a title file which represents the title. This title file is a COS file. Typically in the on-line scenario, this would be a skeleton title. A skeleton title is a COS file which contains only a root moniker and no actual objects. A moniker is an object used in the implementation of the COS and contains identification and status information about COS objects.

A superCOS is a COS file which contains more than one subordinate COS, known as a subCOS. For example, a superCOS at the customer workstation is used to cache objects which have been remotely retrieved from the host data center. As long as these cached objects are not out of date or flushed, the viewer will be able to quickly provide that object the next time it is requested rather than retrieving it from the data center again. This gives the MP system a tremendous speed advantage over other on-line systems.

A top level system flow diagram is presented in conjunction with FIG. **5** and exemplary Viewer screen displays that could be seen during the processes of the system flow diagram are described in conjunction with FIGS. **6** and **7**. An example of the rendering process and a query that are used to display the title to a customer are presented in conjunction with FIGS. **8** and **9**. Finally, a top level software executable and API/DLL view of the system is presented in conjunction with FIG. **10**.

A. Separation of Design and Content in the Multimedia Publishing System

As discussed above, the MPS architecture maintains a clean separation between design information and the content to which that design will be applied. A publisher's collection of page layouts is in the form of one or more titles. A title is a collection of page layouts, in a particular sequence which relates to the order in which pages will be viewed. The page layouts describe how the client area of a window will appear when a page is rendered. Rendering refers to the creation of a bitmap of a display screen in memory prior to displaying the screen. A complete page layout is created by placing controls on a blank page layout, where each control delineates an area where some piece of content should be displayed. Settings on each control determine the proper place to look for the content to be displayed in that control.

The content takes the form of discrete objects, each of which compose one unit of information, e.g., a story or a picture. These content objects are of well-known and public data formats, and may be created using any tool that supports these data formats. Content objects generally do not have formatting information encoded within them.

When the publisher has created the title (with its page layouts) and the content objects, the title and content are published together to the public distribution point. Consumers download the title and content objects to their personal computer, where the MPS viewer software uses the page layouts in the title to compose the content in the visually rich form designed by the publisher.

B. System Configuration

Referring now to FIG. **1**, the basic system configuration of the multimedia publishing system (MPS) **100**, which is a preferred embodiment of the system **100**, will now be described. By convention, the term title is used to describe the overall plan or instructions for assembling the complete on-line MPS application on a customer's computer.

Much of the power of the MP system **100** resides in its ability to fully separate design and content, unlike existing on-line and multimedia publishing tools which require a publisher or content provider, such as a first publisher **102**, a second publisher **104**, or a publisher M **106** to integrate design and content. In the MP system, titles, such as a title A **140**, title B **142**, or title P **144** can be divided into two parts: the content (**148, 152, 156**)—the information such as bitmaps, video clips, audio, animation, or stories that make up a title—and the title layout, also termed the design (**146, 150, 154**)—the overall look and feel of a title. To separate content and design using the MPS rather than placing content directly on a page, a publisher can place the content, such as a set of content objects **112, 114**, or **118**, in one or more containers of a title and then create sections or subsections having pages with special controls, such as a set of title layout objects **110** or **116**, that dynamically find and display the content at runtime.

Using this technique a publisher can change a title on an ongoing basis by merely updating the content **112, 114, 116** which has been placed into various folders or containers within the master title. When a page is displayed, it shows the updated content. This is called dynamic title synthesis or dynamic synthesis, and allows content to be continually updated without any need to modify and update the title design consisting of the individual pages, controls and hand-placed content used to display the content.

When publishers use dynamic synthesis they are creating titles which contain placeholders that will be filled-in by the changing content. When dynamic synthesis is utilized, a title is used as a template and a pressing is the displayed, filled-in

title. Each time the publisher updates the content in a title and makes it available for customers (also known as end-users or client end-users), such as a first customer **160**, a second customer **162** or a customer N **164**, the publisher is creating a new release of that title. When the customer starts to view that release, a "pressing" is made which contains part or all of the content in the release.

A major advantage of this approach is flexibility. Some parts of a title may be created by hand-placing content directly on a page, and other parts may be created using dynamic synthesis. Notice, however, that content hand-placed directly on pages is static—it changes only when the people involved in creating the title update the pages.

Returning to the creation of title layouts and content by the publisher, after creation, the title layouts **110**, **116** and content **112**, **114**, **118** are released and stored in a publication storage **120**. The storage **120** can be implemented in many forms, such as a network **122**, CD-ROM **124**, and other means of storage, such as bulletin boards, magnetic media, cable television and so forth.

The presently preferred network **122** is the Microsoft Network (MSN), which can be accessed, for example, by Microsoft Windows 95. Of course, the MPS is designed to be portable so that it can be used on any on-line network including but not limited to, Internet, America On-Line, Compuserve and Prodigy.

In the presently preferred embodiment of the storage **122** as the MSN, many customers will use a MSN Explorer tool to acquire and activate MPS applications.

The MSN Explorer is the integrated navigation tool within Windows 95 that is also used to browse the MSN hierarchy. Sophisticated customers may use other more advanced MPS features, such as search, scheduling, and automatic delivery, assuming these features have been activated by the publisher. Besides browsing via the Explorer or scheduling automatic home delivery, there are several additional ways customers can obtain MPS applications. For example, an individual application may be distributed via floppy disk or CD-ROM **124**, it may be distributed through E-mail or bulletin boards, or the application may be directly accessible via a link in other applications (such as the Microsoft Network yellow pages system). In each of these situations, the MP system **100** acquires an application for the customer.

C. System Components

Referring now to FIG. **2**, the preferred basic components of the MP system **100** will now be described. The system **100** includes a set of tools for designing, developing and viewing multimedia on-line applications. A publisher, such as the publisher **102**, utilizes a publisher workstation (also known as a computer or machine) **182** and a Designer software environment **194** to create and publish the title layouts **110** and content **112**. In the system **100**, a publisher could possibly just create content and use the title layouts of another publisher. The title layouts and/or content are preferably stored in a network **122** that includes a high-performance server for hosting on-line applications. The preferred network **122** will be further described in conjunction with FIG. **3**. A customer, such as customer **162**, utilizes a customer workstation **182** and a runtime Viewer software component **202** to find and activate MPS titles, stored on the network **122**, on a visual display at a workstation **182**.

The Designer **194** is an extensible design and development environment that includes several preferred software components. These include a project editor **184** to manage tiles, containers, and objects; a page editor **186** to create and layout pages; a style sheet editor **187** to edit style sheets; a search object editor **189** to create search objects; a word processor, such as a MPS Document Editor **188**, for creating content optimized for the MP system **100**; and optional third-party tools, such as a sound editor **190**, an image editor **192**, and another media object editor **193** to create and modify sound, image, video, animation and other content objects. For authoring textual content, the preferred document editor is an enhanced version of the Microsoft Words®6.0 word processing program for creating tagged, hypertext documents. Together, these programs form the Designer Component **194**.

The project editor **184** is used to invoke a style sheet editor **187** that is used to create and edit style sheets. The style sheet editor **187**, and portions of the project editor **184** and page editor **186** will be described in detail in subsequent sections of this discussion.

The MPS Designer **194** is a page or forms-based development system similar to Visual Basic. The development environment is graphical and easy to use. Controls, which represent the components of a MPS application that will appear on-screen, are laid out within MPS pages. MPS pages and controls are preferably based on Object Linking and Embedding **198** (in FIG. **2**) (OLE), Microsoft's component software technology. OLE, which presently is at version 2, is further described in Inside OLE 2 and OLE 2, Programmer's Reference, Volumes 1 and 2, all of which are published by Microsoft Press. In addition, the System Overview chapter of Class Library User's Guide for the MFC Class Library, Microsoft corp., 1993, provides further relevant information. However, other compound document architectures such as OpenDoc could be used as well.

A major feature of OLE is interoperability, the basis for integration between applications. This integration brings with it the need to have multiple applications write information to the same file on the underlying file system. OLE defines a model called OLE Structured Storage for treating a single file system entity as a structured collection of two types of objects; storages and streams. These objects act like directories and files, respectively.

The OLE Structured Storage model generally implements these objects; applications rarely, if ever, need to implement them. These objects, like all others in OLE, implement interfaces: IStream for stream objects, IStorage for storage objects.

A stream object is the conceptual equivalent of a single disk file. Streams are the basic file system component in which data lives; each stream has access rights and a single seek pointer. Through its IStream interface, a stream can be told to read, write, seek, and perform a few other operations on its underlying data. Streams are named by using a text string; they can contain any internal structure because they are simply a flat stream of bytes. In addition, the functions in the IStream interface map nearly one-to-one with standard file-handle-based functions such as those in the ANSI C/C++ run-time library.

A storage object is the conceptual equivalent of a directory. Each storage, like a directory, can contain any number of substorages (subdirectories) and any number of streams (files). Furthermore, each storage has its own access rights. The IStorage interface describes the capabilities of a storage object, such as enumerate elements (dir), move, copy, rename, create, and destroy. A storage object itself cannot store application-defined data except that it implicitly stores the names of the elements (storages and streams) contained within it.

The OLE Structured Storage technology solves problems associated with previous flat file systems through the extra

level of indirection of a file system within a file. With OLE, a particular application can create a structured hierarchy where the root file itself has many substorages. Each substorage can have substorages within it, and so on.

This structure solves the problem of expanding information in one of the objects: The object itself expands the streams in its control, and the implementation of storage determines where to store all the information in the stream.

The MP system **100** includes a number of pre-packaged controls such as navigation controls, rich-text controls, multimedia controls, and other special controls specifically designed to support the creation of MPS applications. Because the MPS is based on OLE, third parties can also design their own controls for use within the MPS (using the Microsoft OLE Control Development Kit that is bundled with Microsoft Visual C++ 2.0). In this way, the MPS development environment is fully extensible so that customers can add new capabilities to their MPS applications by purchasing additional controls from third parties or by creating their own controls. The MPS development environment also includes a Visual Basic for Applications (VBA) scripting and debugging system.

While content is displayed within controls that have been laid out on MPS pages in the MPS Designer **194**, content can be authored in any number of existing Microsoft and third-party tools. One such tool for authoring hypertext is the MPS Document Editor **188** that supports special MPS features for creating and tagging MPS text. Other existing tools for creating bitmaps, complex drawings, and other multimedia content can be used to create the content displayed within any particular OLE Control. In addition, most existing OLE Controls (.ocx executable programs) will work in the MPS environment although they may not be optimized for on-line applications. For example, a standard AVI OLE Control could be placed in an MPS application.

The controls that are part of the MP system **100** are optimized for low bandwidth on-line delivery of data. However, the use of high bandwidth data delivery is within the scope of the present invention. The MPS **100** is designed to operate with information that can change from minute to minute, daily, or monthly. So while the MPS can be used for creating static titles that are hand-crafted and cannot be easily updated on an ongoing basis, the main focus of the MP system **100** is to provide an efficient, cost-effective mechanism to manage the creation and management of dynamic, continually changing on-line applications. At the same time, as an open development environment, many of the tools commonly used for creating static multimedia content can easily be incorporated into the MP system **100**.

When activated by the customer, the Viewer **202** examines the components of a selected title to see if any of the information required to display the pressed title needs to be acquired. It then acquires this information from publication storage **120** or local storage at customer workstation **182** and organizes it so that it can be displayed to the customer **162**. Thus a pressed title captures the set of information that is displayed to the customer at a given point in time. In other words, some titles might produce a new pressing every day, or more frequently, as the content changes. On the other hand, other titles may be static; when a static title is activated there is no need to do another pressing, since the content has not changed.

While pressing a static title may seem unnecessary, the process of organizing and displaying the pressing can take into account customer preferences and display device characteristics. For example, suppose a customer activates a static title on a laptop when using the laptop screen and then later activates the same title when the computer is attached to a larger display. The second activation will result in another pressing to take into account the much larger screen area, if the publisher has enabled such an option. When the title is activated, the MPS Viewer **202** determines if the title is out of date; acquires any needed. information; and then, if necessary, creates and possibly personalizes the pressing.

The MPS Viewer **202** enables customers to perform the following actions within the limits defined by content providers: select and personalize the information a title acquires, modify the overall structural properties of titles, personalize the look and feel of titles, manage and archive the content customers acquire, and view billing and pricing information.

The requirement for the preferred publisher workstation **180** is a Windows 95 workstation with the minimum hardware configuration necessary to run the MSN sysop tools and to store and display the titles under development. The preferred Windows 95 workstation has, at a minimum, an Intel 486 processor running at 33 MHz or better with eight Megabytes of memory. A 9600 baud or faster modem is required to run the MSN sysop tools. For multimedia titles, this includes a MPC2 compliant (multimedia configured) workstation.

The MPS Viewer **202** should be installed on the customer workstation **182** before an MPS title is activated. The presently preferred customer workstation is capable of running Windows 95. To make this installation easy, the Viewer **202** is automatically installed onto the customer workstation **182** the first time the customer connects to MSN and the MP system **100** is enabled. MPS titles may include resources such as fonts, Dynamic Link Libraries (DLLs), and OLE controls that are placed into the resource container or folder of MPS titles. Before customers can view such titles, these resources are installed on their workstation **182**.

D. Network Storage

Referring to FIG. **3**, an exemplary network storage subsystem **122** will be described. FIG. **3** is a high level diagram illustrating the basic components of an on-line network **122** in accordance with one embodiment of the invention. Multiple publisher workstations **102, 104, 106** and customer workstations **160, 164** are connected to a host data center **242** by a wide area network (WAN) **240**. The publisher workstations preferably have high speed connections to the WAN **240**. The wide area network **240** includes WAN lines **244** which are provided by one or more telecommunications providers, and which allow end users (i.e., publishers and customers) over a wide geographic area to access the host data center **242** via modem. The WAN lines **244** preferably include both X.25 lines and ISDN (Integrated Service Digital Network) lines.

The host data center **242** comprises a plurality of application servers **246** connected to a high speed local area network (LAN) **248** (which may include multiple LANs). Each application server **246** has a unique server ID. As shown in FIG. **3**, three of the servers **246** are MP System servers (**246***a*, **246***b* and **246***c*). Also connected to the LAN **248** are multiple Gateway computers **250** also referred to as Gateways, which link incoming calls from end users to the application servers **246**.

It is envisioned that the host data center **242** may advantageously have on the order of one hundred Gateways **250**, and between several hundred to several thousand application servers **246**. A host data center of this type will be able to handle tens of thousands of simultaneous user logon sessions.

As described below, the server side of each on-line service is preferably implemented using one of the following: (1) a

single application server **246**, (2) a set of "replicated" application servers (i.e., application servers which run the same service application or applications) that provide access to replicated (and locally-stored) copies of service "content" data (i.e., data provided to end user's of the service), or (3) a set of replicated application servers that provide access to server-specific (non-replicated) service content data.

The host data center **104** also includes multiple Arbiter computers **252** that monitor, record and process certain types of transactions to ensure consistency among replicated application servers. The host data center **104** also includes one or more custom Gateway computers **254** which link the host data center **104** to one or more external service providers **256**, such as a credit card service that validates and executes credit card transactions.

The host data center **104** also includes a number of administrative servers **258**. The administrative servers **258** perform administrative functions such as accounting, billing, network management, backup, system security, performance analysis, and server-to-service allocation.

To route user service requests to the appropriate servers **246**, the Gateways **250** must have some way of determining the unique IDs of the servers that are currently handling the requested services. This is accomplished by means of a service map (not shown), which contains information about every service and server **246** in the host data center **242**.

The service map is preferably generated by a service map dispatcher **260**, which may be implemented on a single computer.

In addition to generating a service map, the service map dispatcher **260** maintains a central repository of information referred to as the "global registry" **262**. The global registry **262** contains various information about the present configuration of the host data center **242**. For example, for each service group, the global registry **262** indicates the IDs of the servers **246** of a service group, and the identity of the Arbiter computer **252** (if any) which is assigned to the service group.

Further disclosure of the preferred network **122** is provided in a copending application also assigned to the assignee of the present application, Microsoft Corporation, entitled "Architecture for LAN-Based On-Line Services Network", U.S. Ser. No. 08/503,307, filed on Jun. 7, 1995.

E. Container Hierarchy

Referring now to FIG. **4**, the high level hierarchy of containers for a plurality of publishers using the MP system **100** will be described. In the presently preferred embodiment, the MP system **100** utilizes a specific directory structure with the MSN directory tree. This structure is rooted at a specific folder (specified via the MSN global registry **262**) known as a container of publishers **280**. Every publisher **102**, **104**, **106** will have at least one container or folder called a project. For example, the publisher **102** has a folder called Project A **282**, the publisher **104** has two folders called Project B **284** and Project C **286**, and the publisher **106** has two folders called Project N-1 **288** and Project N **290**. Content folders and/or titles are dropped into the folder of the publisher.

Allowing for multiple projects satisfies the needs of a large publisher. For instance, a project could be assigned to one magazine (e.g., gardening) and another project could be assigned to another magazine (e.g., motorcycling). Thus, each month's issue could be archived as a title according to volume and number in its respective project.

As an example of how projects could be configured, Project A **282** only has a content folder **292**; Project B has a title folder **294**, and two content folders **296** and **298**, along with a link to the content folder **292** of publisher A **102**; Project C has two title folders **300** and **302** that could share a content folder **304**; Project N-1 has a title folder **306** and a content folder **308**; and Project N has a title folder **310** and shares content folder **308** with Project N-1. Publisher **102**, for example, could be a provider of raw statistics in content folder **292** but does not want to generate title layouts. The publisher **102** may have an agreement with the publisher **104** for the publisher **104** to allow access and use of the content in the content folder **292**. The publisher **106** has two projects **288** and **290** that share the content folder **308**, for example, due to the common subject matter of titles in title folders **306** and **310**. As illustrated in FIG. **4**, a project, such as the project **286**, may contain multiple titles folders.

F. Operational Description Including Top Level Flow Diagram

Referring now to FIG. **5**, a top level flow diagram of the processes performed using the MP system **100** will now be described. The flow diagram and this description introduce the process **320** a publisher **102** or information content provider (ICP) would use to design and distribute MPS titles.

As previously stated, a title is a publication, application, or service created using the MP system **100**. For example, the publication may be a monthly magazine, wherein each issue of the magazine is a new title. A title consolidates the set of instructions for assembling the information that is displayed to the customer **160**. Customers see titles as icons on the Microsoft Network, on CD-ROMs, or in a file system. By double-clicking (activating) on the title, name or icon, the customer can interact with the title.

1. People and Tasks involved in Title Creation

The MP system **100** is designed to support large teams creating complex on-line applications, as well as small teams creating individual works (and anywhere in between). This section, however, discusses only the more complex, high-end operations. In simpler scenarios, one person could perform more than one of the roles described below, and the amount of materials (stories, artwork, advertisements, and so on) would be more limited than the materials described here.

The process of creating and publishing a MPS title can be broken into a title-design phase and a release-creation phase. The process is set up so that all of the content and layout that is common across releases can be performed once in the preparatory design phase, and then left alone. This allows for a smaller team and faster turnaround in producing each release.

a. Title Design

The process of creating a new title begins with the editor. Assisted by business development staff, the editor decides on a target customer base, and on a concept for the title that will appeal to that base. This design team then develops that concept into a proposed organization for the contents of the title.

Before content can be put in place, a framework for the title must be created. This involves:

- Creating a section hierarchy within the title.
- Creating content folders to store stories, advertisements, and other pieces of content.
- Creating search objects in each section of the title that draw content from the appropriate content folders using specified criteria.

In some organizations, this work will be done by the editorial staff. In others, it may be done by the production staff.

Once the basic framework is in place, the art department can create artwork to fill in the title's common elements. This includes:

- A style sheet describing font usage and text layout.
- Page layouts for sections that dynamically gather their content.
- Page layouts for sections that are always the same (cover, title pages, mastheads, and so on)
- Logos.

Optionally, organizations may want to include developers in the title design process. For example, the particular application being designed may benefit from the use of custom designed OLE Controls. These controls could be purchased, or developed in-house using the Microsoft Visual C++ development system. Additionally, the advanced features of the Blackbird system, including accessing the API or scripting controls to respond to events or automatically perform actions at runtime would require some development work, either in the high level scripting language (VBA), or in a lower-level language such as C++.

b. Authoring and Title Release

Once the framework is created, the staff can now turn their attention to creating individual releases. All of the work done in the conceptual phase above is potentially re-usable for every release. In fact, for a title with little need for detailed artwork, the rest of this process could merely be a matter of dropping edited content (including advertisements) into content folders.

For dynamic titles, most (and potentially all) of the work is done within the Content Authoring environment. For static titles, it could all be done within the Title Design environment. In practice, most releases will involve some work in both of these environments.

i) Writers Provide Tagged Content

Content authors—including editors, writers, reporters, and forum managers—generate content, including structured stories, using the content authoring environment. Writers compose the textual content that appears in a title (or a release of a title). They hand their materials off to the editorial staff. The editorial staff is in charge of the overall content of the title. For multimedia titles, this role is very similar to the director of a motion picture or television program.

The content authoring environment supports a variety of tools, such as, for example, a MPS document editor. The MP system **100** also supplies tools to specify and manage links and to specify story properties. Third-party tools may also be added to the content authoring environment.

From a content author's perspective, creating structured stories can be as simple as typing them in the MPS document editor and applying certain styles. More sophisticated content can be created though a variety of means, such as including links to graphics or placing special properties on a story.

For content providers that do not want to expend much effort creating tagged content, the MP system **100** includes MPS document editor templates that handle most of the tagging for the author.

ii) Editorial Staff Chooses Content

Once the editorial staff has chosen the stories they wish to include in a release and are satisfied with the content of those stories, they pass them on to the art department to select and insert appropriate artwork, and to the production staff to place in content folders.

iii) Art Department Supplies Specific Art

The artistic staff is responsible for designing the more graphical aspects of the title. In the early conceptual phase, graphic artists work with the editor to design a distinctive look and layout. This includes font styles, colors, titles, logos, and page layout templates. The term "art department" is used in the broadest sense here. In the multimedia world, the role of an art department goes beyond traditional print-based artwork.

The art department in many cases inserts the artwork into the stories and tags that artwork so that it will presented appropriately (placed inline in the story text, as a wrap, or as a pop-up). They then pass the stories on to the production staff to be placed in content folders. In the case of static titles, the art department designs new pages and gives them to the production staff to be placed in the title framework.

iv) Advertising Department Supplies Copy

The advertising sales staff sells advertising space in each release. The advertising sales department collects copy from advertisers who have bought space in the release, and delivers the copy to the production staff to be placed in content folders.

v) Production Department Does "Paste-up", Proofing and Release

The production staff does the fundamental tasks, such as paste-up, necessary to put a title or release together. Once the production staff has everything that goes into the release, they "paste up" the release by placing everything in its appropriate place and performing a "test-pressing" to make sure that nothing is missing. The editors, art staff, production staff, and advertising staff review the test-pressing to make sure that everything looks and works correctly. Once everyone is satisfied, the production staff places everything on the publisher's server and releases it to be copied to additional servers at the Microsoft Network data center.

2. Top Level Flow

The process **320** begins at a start state **322** and continues at a state **324** wherein the publisher **102** uses the MPS project editor **184** (FIG. **2**) to create a project on their workstation **180**. A project, such as project C **286** (FIG. **4**) contains all the information needed to build and distribute one or more titles and any associated content.

Moving to state **326**, within the project, the publisher **102** creates titles and content folders, such as title **300** and content folder **302** (FIG. **4**). A title consists of nested sections that contain MPS objects such as pages or search objects. Folders typically contain MPS content objects such as stories or pictures. To make the process of managing titles, folders, and MPS objects easy to understand and use, the preferred MPS project editor **184** (FIG. **2**) looks and works like the Windows 95 Explorer.

Proceeding to state **328**, the publisher **102** uses the MPS project editor **184**, page editor **186**, style sheet editor **187**, and search object editor **189** (FIG. **2**) to create the MPS layout objects such as pages, styles, and search objects. The page editor **186** is also used to place controls (each control is a program responsible for handling a displayable region) on a page.

Moving to state **330**, the publisher **102** creates content objects using the MPS Document Editor **188**, or the publisher can use third-party tools, such as the sound editor **190** or the image editor **192**, that produce formats that the MP system **100** can interpret. The authoring and processing of content objects is further disclosed in a copending application also assigned to Microsoft Corporation, entitled "Structured Documents in a Publishing System", U.S. Ser. No. 08/503,307, filed concurrently herewith.

The creation of content objects could also be done prior to any of states **324**, **326**, or **328**. After the content objects are created at state **330**, the publisher invokes the page editor **186**. If not previously done at state **328**, the publisher lays out each page with at least one control. Selecting a control on a page lets the publisher bring up a context menu, of

which one item is a Properties selection. Choosing the Properties selection brings up a control's property sheet. Among the property sheet pages are a story page and a picture page. The story page allows the publisher to choose a story content object that is to be displayed in a story control. The publisher could enter a path name to the desired content object. Alternatively, pressing a ". . ." button brings up a Content Browser dialog which allows for browsing within the project to find a desired story content object. The picture page is used for choosing a picture object to display in a control. The publisher could enter a path name to the desired content object. Alternatively, a Content Browser dialog allows the publisher to choose a picture content object from within the project. Other types of content objects are associated with a layout object in a similar way. Further descriptions of the property sheet pages are provided below in conjunction with a discussion of controls.

Proceeding to state **332**, the publisher **102** releases the project. In the presently preferred embodiment, releasing a project makes the titles, stories, and other MPS objects available on the Microsoft Network **122**. The MP system **100** automatically connects to the network **122** and makes the titles in the project available to the customers **160**, **162**, and **164** (FIG. **1**). Alternatively, the MP system **100** can release the title to CD-ROM **124** or other storage/communications media.

Continuing at state **334**, the customer **160** uses the MPS Viewer **202** (FIG. **2**) to read and page through (also termed navigation in an electronic publication) the released titles. As parts of the title are accessed, they are cached on the customer's computer **182** for fast access. The viewer **202** organizes and composes the objects it has collected and displays them to the customer **160**.

Over time, the publisher **102** can update the project and the MP System automatically tracks the changes. Decision state **336** determines if the publisher desires to update the project. If the publisher does not wish to update the project, process **320** completes at end state **338**. However, if decision state **336** is true, that is, the publisher desires to update the project, the process **320** moves to a decision state **340** to determine if the publisher **102** desires to modify the layout in the project. If so, the process **320** moves to state **342** wherein the publisher modifies one or more existing layout objects or adds one or more new layout objects. If the decision state **340** evaluates to be false, or at the completion of state **342**, the process **320** moves to state **344** wherein the publisher modifies or adds one or more content objects. At the completion of state **344**, process **320** proceeds to state **332** wherein the project is released again. Releasing the updated project ensures that the proper set of layout and content objects are made available to the customer **160** (FIGS. **1** and **2**).

G. Exemplary Screen Display of Title

Referring now to FIG. **6**, an exemplary screen display **360** of a page of a title as displayed by the Viewer **202** on the visual display at the customer workstation **182** (FIG. **2**) will now be described. The screen display **360** corresponds to a World News section of a MSNLive title using a page layout which has been named NewsFront by the designer. A tabbed horizontal bar **362** near the top of the screen **360** is handled by a caption button control and shows the major sections of the title. By selecting a section name (by use of a pointer device like a mouse, not shown, but which is a part of or connected to the workstation **182**), the customer **102** can navigate directly, through a link, to the selected section.

Below the bar **362** of screen **360** are two headlines **370** and **372** which are the result of an outline control that can be used as links to corresponding stories on another screen of the title. Block **373** in this example contains an advertisement resulting from a picture control. Block **374** contains a graphic and text resulting from a picture button control that provides a link to a weather screen. Areas **380** and **384** display headlines for corresponding abstracts **382** and **386**, respectively, and are the result of an outline control. By selecting the headline **380** or **384**, the customer can navigate to the body of the corresponding story on another page of the title. Areas **390** and **392** display picture objects corresponding to the headlines **380** and **384**, respectively, and are the result of picture controls.

The objects and placement of the objects on the displayed page **360** are determined by the publisher **102**. Of course, other objects or placements of objects could be utilized by the publisher **102**.

H. Exemplary Screen Display of Project Editor Window

Referring now to FIG. **7**, an exemplary screen display **400** of the parts of the content and layout for the example title displayed in FIG. **6** will be described. The Project Editor window **400** is the main interface for the Designer **194** (FIG. **2**). The window **400** is intended to closely mimic the Microsoft Windows 95 Explorer. Using this window **400**, the publisher can open, edit and save a project, as well as release the contents of that project to the MSN Data Center **242** (FIG. **3**). An approximately left one-third of screen **400** is a display area **402**, also known as a left pane, that shows the hierarchy of containers of one project for a publisher and allows the user to navigate through it. The left pane shows only containers (folders, titles, and sections). An approximately right two-thirds of the window **400** is a right pane **404** that shows the contents of a container selected in the area **402** by the user of the project editor **184** (FIG. **2**).

Referring to the left pane **402** of the window **400**, the top level of the hierarchy of containers is the project "MSNLive" **406**. Just below the project is the title "MSNLive" **408**, which in this example has the same name as the project **406**. In another example, the project could have a plurality of titles, such as a January issue of a magazine "X", a February issue of magazine "X", and so forth. Below the title in the example hierarchy are two sections: "News" **410** and "Sports" **414**. Also at this level in the hierarchy is a content folder **418** labelled "Graphics", which holds the picture objects used by the project **406**. Below the sections **410** and **414** is a set of subsections **412** for the "News" section **410** and a set of subsections **416** for the "Sports" section **414**. The "News" section container **410** has been selected by the user, which is evidenced by the highlighting of the section label "News" and the opened section icon to the immediate left of the "News" label.

Referring to the right pane **404**, the layout objects and content objects directly contained within the selected container in the left pane **402** are shown, e.g., the objects of the "News" section container are displayed in this example. The left pane **404** uses standard Explorer views, as well as a special view built for the window **400**, which sorts according to a user-defined order and allows the user to change the order by dragging and dropping each objects' icon. The objects are preferably grouped by type of object, such as, for example, subsection objects **412**, page layouts **420** and content objects **422**. The order of the pages and content objects is significant. The title maintains a sequence ordering of the sections, pages, and search objects, as this is important in determining how the title is displayed. Within a section, the pages have a sequence that determines the order in which they are used to press content and the order in which they are displayed when the user browses sequentially. In a static

section, pages are displayed in the order shown in the project editor window **400**.

A dynamic section uses the dynamic story control (FIG. **8**) to display stories within a section. The stories are sorted according to rules specified on the section's property sheet and then are concatenated or linked together. The stories are then filled into the dynamic story controls on each page in the section, in the order in which the pages are arranged in the section. If there are more stories than there are pages, the last page is re-used repeatedly until all content has been pressed. For instance, in FIG. **7**, the Backpage in pages **420** would be reused.

Toolbar buttons and corresponding menu commands allow the publisher to quickly add new objects to the titles and folders within the project **406**. Clicking a button will add a corresponding object to the container selected in the left pane **402**. Only those objects that are allowed to be in the selected container have their corresponding toolbar buttons and menu items enabled.

I. Example of Rendering Process

Referring now to FIG. **8**, the interaction of page layouts, having controls, and objects at the Viewer **202** (FIG. **2**) of the customer's workstation **182** to render pages will now be described. The Viewer **202** supports the display of information through windows. The placement, organization, and number of windows is under the control of the publisher **102**. Viewer windows are Windows 95 frame windows. These windows are completely under the control of the designer. The designer controls the Viewer **202** by creating a title. The title sets the size and standard elements (title bar, Min/Max buttons, caption, border, menu bar) of the various windows displayed by the Viewer **202**.

The entire client area of a viewer window is used to display a series of pages. Each page contains a set of controls that are used to display content, to navigate through the title, and to gather information from the customer. In response to customers actions or other events, the page that is displayed may change during the course of running the title. This behavior is determined by the publisher **102**. A title may have more than one window visible at any given time, and popup windows may be modal or modeless. Only one title may be displayed within a Viewer window at any given time.

FIG. **8** presents a diagram of a front page section **430** and a business section **432** for a title, such as a newspaper.

1. The Front Page Section

The front page section **430** contains a page **434** which has a picture control **436**, and a set of static story controls: a first story control **438**, a second story control **440**, and a third story control **442**. Each static story control or picture control is linked at publication time to just one object. Each of the controls on the page **434** references a style sheet **443** to provide formatting instructions on how the content is to be displayed.

As shown in FIG. **8**, a picture object **460** is linked to the picture control **436**, so that upon rendering, the picture object **460** is displayed on the page **434** at a position determined by the control **436**. Similarly, a story object **462** is linked to the static story control **438** and rendered into the position of the control **438** on the page **434**.

Note that since the control **438** is a static story control, any area not used by the story object **462** in the area identified by the control will be blank. As shown, a story object **464** is linked to the story control **440 80** that it is rendered in the area identified by the static story is control **440** on the page **434**. In this example, for instance, only the first paragraph of the story object **464** will be rendered on the page **434** due to the size of the control **440** (as selected by the designer). In this manner, the designer can choose to only display a portion of a linked story within a static story control by adjusting or sizing the control to only hold one paragraph, or other desired portion, of the story content. Normally, a static story control will allow scrolling of a story so that ultimately the entire story will be displayed.

Finally, a story object **466** is linked to the story control **442** so that it is rendered in the area identified by the static story control **442** on page **434**. In this example, the entire story object **466** is rendered onto page **434**.

It is important to note that each of these story objects makes reference to the style sheet **443** before being rendered on the page **434**. When story objects are authored, they are given formatting tags that represent specific styles. As the story objects are rendered, they reference the style sheet that is linked to the appropriate control to retrieve formatting information. This formatting information includes properties of the paragraphs, fonts and embedded objects in the story that format the content as it was originally designed. Due to the separation of design and content in the MP system, the story objects themselves only have formatting tags, but do not contain a description of the particular format that corresponds to each tag. The descriptions of those tags is found in the style sheet that is linked to the control into which the story object becomes rendered. This process will be explained in more detail below with respect to FIGS. **9–15**.

2. The Business Section

As also shown in FIG. **8**, the business section **432** contains a first page **444** and a second page **446**. The page **444** has a single static story control **448**, a single picture control **450**, and a first dynamic story control **452**. The second page **446** has two dynamic story controls, **454** and **456**. In addition, a style sheet X **457** and a style sheet Y **459** are referenced by the different controls on pages **444** and **446**. The pages in the business section **432** differ from the page **434** in the front page section **430** because they rely on a search object **468** to retrieve particular stories. On the page **434**, the static controls were each linked to a particular story which was then displayed upon rendering. The search object **468** is affiliated with the dynamic story controls in the section **432**.

As shown in this example, the static story control **448** and the picture control **450** on the page **444** reference or link to the story object **464** and the picture object **460**, respectively, and display these objects as shown on the rendered page **444**. The story object **464** is thereby shared between different sections, pages and controls in the title. The entire story object **464** is displayed on the page **444**, whereas only the first paragraph was displayed on the page **434**. By using a similar process, a designer can choose to display just the first paragraph of a story on the first page of a title, but include the entire story on another page within the same title. As shown in FIG. **8**, the picture object **460** is also shared between the control **436** and the control **450**. This sharing of content between separate sections and pages is an important feature of the MP system **100**.

3. Dynamic Story Controls

The dynamic story control **452** uses the results of a query performed by the title to retrieve stories matching search criteria set by the publisher (as defined by the search object **468**). The search object **468** locates story objects having specific properties. In the example of FIG. **8**, the search object **468** returned many story objects **470**, **472** and **474** corresponding to story objects 1 through N, respectively (where Nx4 in this example). All of the retrieved story objects are concatenated together by the dynamic story controls and poured into the appropriate regions on the pages. The order that the stories become rendered into the

control regions starts with the first dynamic story control on the page in the section and continues to other dynamic story controls contained within the section.

If enough pages to display all the located stories are not defined in the section, the last page used is repeated until all stories are rendered. Thus, the first located story **470** is poured into the area defined by the dynamic story control **452**. Since it does not completely fit in that area, the located story **470** continues across the page boundary onto page **446** into the area defined by the dynamic story control **454**. The located story object **472** then begins after the located story object 1 **470** ends. The next located story object (located story object 3) begins after the story object. **472** ends, continuing into the next control **456** on page **446**, as shown in this example. The last located story object **474** retrieved by the search object **468** in this example is then rendered into the dynamic story control **456** within page **446**.

As explained above, the dynamic story controls in the section **432** use the search object **468** to display the results of queries made for specific information. For example, the search object **468** may return content that contains the word "Microsoft". Each of the stories found by the search object **468** will be displayed in the areas defined by the dynamic story controls in the format designated by the style sheet **457** or the style sheet **459**.

For example, if the dynamic story control **454** is linked to the style sheet **457**, then all of the stories displayed by the dynamic story control **454** will appear in the format designated by the style sheet **457**. However, the stories rendered by the dynamic story control **456**, when this story control is linked to a different style sheet (for example, the style sheet **459**), would appear differently than the formatted display corresponding to the dynamic story control **454**. In this example, if the controls **454** and **456** use different style sheets, the located story **3** would be displayed using two formats when the transition from the area defined by the control **454** to the control **456** was made.

J. Style Sheet Overview

Style sheets and the style objects they collect are created by the designer (i.e., the person at the publisher workstation **180** shown in FIG. **2**) using the Project Editor and the Style Sheet Editor. Once the style sheet has been created, it is stored in the cached object store (COS) along with the other objects in the project as described above in reference to FIG. **2**. The style sheet objects support OLE serialization and are therefore based on the Microsoft Foundation Class (MFC) Cobject class. These class definitions are publicly available from the assignee.

As described at the beginning of the detailed description section, a different style sheet may be linked to each control region on a page. However, in all likelihood, style sheets will be shared among more than one control. As is known in the present software technology, a globally unique identifier (GUID) can be used in OLE object-oriented environments to identify an object with a unique string of characters. Normally, unique GUIDs are produced by concatenating the time, date and network card serial number of the computer at the time that the object is created. By using this method, it is virtually impossible for two objects to receive the same GUID. In the MP system **100**, each control keeps a record of a GUID associated with its linked style sheet. This is how a particular control can reference its linked style sheet. More than one control can refer to the same GUID, and therefore share style sheets. When a control needs access to its associated style sheet, the control requests the style sheet from the title. If the style sheet has not already been loaded into volatile memory, the title object handles loading it from the COS.

K. Customer Query

Referring now to FIG. **9**, a query path from a "Find" dialog through title searches to the content database at the publication storage **120** will be described. The query components for both publishers **102** and end-user customers **160** are defined as follows: a MPS Document Editor Summary Information dialog—for tagging content with keywords to aid in retrieval; a Search Object Editor—for title designers to create and modify search objects (also known as information magnets); and a Find dialog **510**—a customer interface for ad-hoc and saved searches.

Content, such as stories **502**, **504** and **506**, is tagged using the MPS Document Editor's Summary Information dialog and is placed in the MPS content database in the publication storage **120**. Search objects gather stories which match a particular criteria (as defined in the Search Object Editor) and "flow" them into the appointed sections of a title in the Viewer **202** (FIG. **2**). The Search Object Editor is the query tool which designers use to retrieve and locate relevant stories within the title. The customer **160** uses the Find Dialog **510** within the MPS Viewer **202** to issue one or more queries **512** against all the stories in a particular title (i.e., those stories the title has retrieved using one or more search objects).

The queries **512** issued by the customer **160** in the Find dialog **510** are joined with the criteria of the title's searches due to the search object(s) and then the aggregate is queried against the content database in the publication storage **120**. Result GUIDs **514** (representative of stories matching the queries and search objects) are transmitted back to the customer and appear in a results pane **516** of the Find dialog **510**. By combining the query **512** with the search object queries restricts the results to be within the title structure rather than from any arbitrary source in the content database.

L. API/DLL View of System

Referring now to FIG. **10**, the major software components or modules used by the presently preferred implementation of the MP system **100** will be described. The modules are located at the publisher location **102** (also shown in FIGS. **1** and **2**), at the network storage location **122** and at the customer location **160**.

The modules at the publisher location **102** include a publisher executable **530**, a set of publisher DLLs **532**, a set of publisher OLE custom controls **534**, a publisher COS **544** with a client object broker service and client publisher interface **546**, OLE **198** and MFC **562**.

The modules at the customer location **160** include a viewer executable **538**, the set of common publisher DLLs **532**, the set of common publisher OLE custom controls **534**, a viewer COS **548** with a client object broker service **550**, OLE **198** and MFC **562**.

The modules at the storage location **122** include a server executable **536**, and a server superCOS **540** with a server object broker service and server publisher interface **542**. The publisher executable **530** (also known as BBDESIGN.EXE) is an application which provides a mechanism for generating a design-time view of a project. It is utilized in the creation of objects within a project, and for establishing the relationships between the objects of a project.

The set of publisher DLLs **532** includes a forms DLL (FORMS3.DLL) that provides the implementation of the OLE Control Container class and supplies the data for the page object in a project. Also included is a view DLL (VIEWDLL.DLL) that provides a set of MPS Object definitions and the viewer engine for synthesizing the run-time view of a title. The MPS Objects include: CProject, CTitle, CSection, CFolder, CContentFolder, CRootContentFolder,

CProxyTable, CContent, CFrame, CBForm, CVForm, CStyleSheet, and CMagnet.

The set of publisher OLE custom controls **534** (also known as BBCTL.OCX) is a DLL which provides the code for implementing instances of the OLE custom controls which are standard for the MP system **100**.

The viewer executable **538** (also known as BBVIEW.EXE) is an application which provides a mechanism for initiating the run-time view of a title. It uses the publisher OLE custom controls **534** and the publisher DLLs **532**, especially the viewer engine for synthesizing the run-time view of a title.

Each of the publisher **102**, customer **160** and network storage **122** locations has a COS implemented by using a DLL (COS.DLL). The COS DLL provides a persistent storage mechanism for objects used by the MP system **100**. The COS DLL uses OLE Storage technology to store sets of objects in a compound document file. Each object placed into a COS is given a unique identity, referred to as a GUID. Each object identified by a GUID can be located independent of a path name in a file system. The server executable **536** (also known as MSNSERVER.EXE) is an application which provides a mechanism for managing the network server, which includes the COS. In addition to the COS DLL, the server has a DLL for COS access and object binding (OBSV.DLL), a MPS server service (BBOBSVC.DLL) and a memory management service (DFARBSV.DLL).

Each of the publisher **102**, customer **160** and network storage **122** locations has an object broker service DLL (OBJBRK.DLL). The object broker service attempts to locate an object given its unique identity (GUID). By default, the object broker first looks in its local object store (referred to as a superCOS), which is either the publisher COS **544**, the server COS **540** or the viewer COS **548**. If the object is not located at the COS wherein the request was made, and if the object broker resides on a client machine (either the publisher or customer workstation), it will attempt to remotely retrieve the object from the server COS **540** at the MSN Data Center **242** (FIG. **3**). In another embodiment, other object stores may register with a given object broker as a source of objects, which the object broker will search in between the local and remote retrieval cases. Associated with the object broker **546** at the publisher is the client side of the publisher interface, and associated with the object broker **542** at the network server is the server side of the publisher interface. The publisher interface is used to manage the publication of new, deleted, and modified objects.

The capabilities of the object broker allow a publisher to test layouts or content that are shared with a different publisher. As an example, publisher A has a title layout A and publisher B has content that publisher B has agreed to share with publisher A. To test title layout A together with the content, publisher A could retrieve content provided by publisher B that is stored in the COS **540** by use of the object broker service.

A MPC Wrapper DLL (MWRAP.DLL) uses the Microsoft Network Procedure Call (MPC) protocol to communicate with the network storage **122**, i.e., the MSN Data Center **242** in the presently preferred embodiment, and the MPS services, such as the object broker and COS. This wrapper specifically isolates the COS/Object Broker subsystem from the specific MPC protocol so that the MP system **100** can be easily ported to use other protocols in other embodiments.

IV. DESIGNER ENVIRONMENT

This section of the detailed description will describe the designer environment at the publisher site. The concepts of sharing content among titles and of separation of design and content will be more fully described. This section begins with a discussion of the presently preferred authoring subsystem used by the MP system **100**. Then, a title designer subsystem will be described, which includes the objects available to the title designer; the project, page, style sheet and search object editors used to create and revise the layouts; and the controls used to define the layout of a page. Next, the architectural structures used by the system to enable the creation, revision, and storage of design layouts and content will be described. Finally, the operation of the designer process and release process will be discussed.

A. Authoring Subsystem

Content is separated from design in the MP system **100**. In the Viewer **202** (FIG. **2**), content and design are brought together by controls to display a title as specified by the designer. As a result, these controls need to identify different elements in the structure of the content so they may format it correctly. This is done by creating structured content. The MPS authoring environment provides a way for authors to create structured documents.

The MPS authoring environment includes the MPS Document Editor **188**, which supports the creation of structured documents, insertion of links and the application of properties to these documents for content retrieval. The MP system **100** uses SGML (Standard Generalized Markup Language) to define the scheme for creating structured documents. SGML is a standard for defining a markup language—a set of tags and attributes used to identify the structure of a document called a DTD (Document Type Descriptor). The MPS Document Editor **188** will support saving documents in a format which conforms to the MPS DTD (MPML—Multimedia Publishing Markup Language). This DTD will be published for use with other SGML authoring systems. As part of this environment, the MPS provides a pair of Document Editor converters for reading/writing MPML files, a template defining styles and macros used to create MPML files along with some OLE controls used to insert links and apply properties to these files.

To create content for the MP system **100** in the MPS Document Editor **188**, an author creates a document based on the MPS template. This template provides a set of predefined styles along with supporting macros. The author applies these styles to the text to identify the different elements of the document (headline, abstract, body text, and so forth). Only the predefined styles should be used. When the document is saved in MPML format, these styles are mapped to SGML tags by the MPML output converter. The result is a tagged document which can later be parsed by the Viewer **202**.

The MPML converters for the Document Editor **188** support mapping styles applied to the text to MPML tags. In addition, it will support graphics inserted with the "Insert Picture" command of the Document Editor **188**. This will support both linked and embedded graphics and tag them appropriately. The converters also provide support for the MPS OLE controls provided to insert links and apply properties to the documents.

An important aspect of the authoring environment is that it is only to be used to generate tagged content. The author should not expect that the style definitions made or formatting applied in the Document Editor **188** will carry over to the Viewer **202** when the document is displayed.

As part of the authoring environment, several OLE controls are provided which interact with the MPS environment to help the author insert links and apply properties to documents. These controls are normal OLE objects which

are extended to support rendering their data in MPML format. The MPML converters will be able to recognize OLE controls embedded in the Document Editor document and ask them for their MPML representation using a well-defined interface. When the converters encounter an OLE object, they will attempt to retrieve a MPML representation from them using this interface and insert it into the output MPML stream. OLE controls which do not support this interface will be ignored. The use of the interface allows extending the authoring environment with new OLE controls as needed.

1. Story Editor

A MPS story editor, which is part of the MPS Document Editor **188**, is the main tool designers and authors use to create MPS story objects. A MPS story object consist of a stream of text with embedded objects such as links or pictures. MPS stories can also be tagged with Find properties so that the MPS Find system can easily and accurate locate stories.

The main tasks involved in the creation and delivery of a story are: author the story; set structural properties for the story; optionally, place pictures into the story; optionally, create links to other stories, and set summary properties (including Find matching criteria) for the story.

In addition to using the MPS Document Editor **188** to create stories, publishers can create MPS stories with other tools or with an automated process. For example, stock ticker stories probably will be created automatically.

MPS stories are structured, which means that the elements that make up the story are logically identified. This is useful because the MP system **100** can take advantage of this logical description to help present the information to users. The Document Editor **188** makes this easy, wherein authors merely apply the Document Editor styles. This process may also be performed automatically using filtering software that is supplied by Microsoft or by third parties.

The MP system **100** supports three main file formats. These are: (1) the MPS data file format, (2) MPML, and (3) the HyperText Markup Language or HTML. The MPS data file format is the native MPS story format. It is a standard OLE doc file with separate streams for text and the various objects contained within the text stream. The MPML format is available to make it easy to import and export MPS stories. A MPML file is an ordinary text file that conforms to SGML. Because this file format is a simple text file, it is easy for publishers to automate the process of creating MPML files. Most publishers will not need to use MPML because the MPS tools automate the process. The MPML file format is only important because it can be easily converted to other formats, which ensures an easy migration path for publishers.

The MP system **100** can also import and export HTML text files. However, because HTML is fairly limited many advanced MPS features can not be represented in HTML. The HTML and the MPML converters are constructed as a separate program that enables publishers to make batch translations of files.

Stories are usually linked to other appropriate content, and MPS Find properties are added to the story so the story can be found by the query subsystem. These steps can be performed using MPS or third-party authoring tools. If a publisher uses third-party tools to produce content, the results must conform to the MPS file formats to ensure that the Viewer **202** can interpret the content.

2. Links

MPS stories typically have links to other stories or other information. The MP system **100** supports these hyperlinks through a link editor. The link editor is integrated into the Document Editor **188** and is accessible from the toolbar buttons or from an Insert\Hyperlink menu. A content selector is used to select the target of links and to select pictures to embed in MPS stories.

3. Find Properties

To help customers find stories that might be interesting, the MPS supports the specification of keyword or keyphrase matching criteria through the file summary information option. A standard File\Summary Info dialog of the MPS Document Editor **188** is used to tag a story with retrieval attributes for search to find. Each field may be individually searched by the search editor. The Find dialog may search the title field uniquely, but the rest of the fields (subject, author, keywords, comments) are searched as a whole when the 'Summary' box in the dialog is selected.

B. Title Designer Subsystem

1. Overview

This section describes the MPS title design environment, with emphasis on the Project Editor tool **184** (FIG. **2**). The Project Editor **184** is the tool that provides a view into a project, and allows the designer to edit the contents of that project.

A Source is a collection of MPS story objects stored on the MSN **122**. In the presently preferred embodiment, each content folder is a separate source. In another embodiment, a single source may contain multiple content folders.

2. Objects

The Title Designer may interact with an open and extensible set of objects that are placed within a project (either in a folder or in a title). Objects appear in the Title Designer just as documents appear in the Windows 95 Explorer: as icons in the right pane. Objects respond to clicks, double-clicks, right-clicks (which display a context menu), and drag-drop operations as means to manipulate them.

Nearly all objects in the system have a property sheet. This is a standard-format dialog which allows for setting values associated with that object. It is accessed by selecting Properties . . . from the File menu, or from the context-menu of the object. A property sheet consists of one or more tabbed pages, which the user may flip through by clicking on the tabs. At the bottom of the page are three buttons: OK, Cancel, and Apply. OK saves any edits that were made and dismisses the dialog; Cancel discards any changes and dismisses the dialog. Apply saves any edits that were made, but does not dismiss the dialog. Switching between tabs does NOT save changes that were made to the previous page.

a. Project

A project is a collection of titles and content folders. Titles and content folders are collected so they can be edited simultaneously and then released in the preferred embodiment to a MPS server **246** (FIG. **3**) together.

A project object represents the entire contents of the project, and is the container of things that get released together. As such, it has properties representing where the project's contents are released to, and statistics about the release process.

b. Title

A title is a collection of pages and supporting objects organized into a hierarchy of sections. The title itself, as well as each section, acts like a folder in that it can be expanded to show its contents within the Title Designer. It is important to note, however, that the items within a title have a hierarchical structure that is defined by the designer and is essential to how the title is displayed.

A title also has a Resources folder that contains any additional objects that need to be distributed with the title;

for example, fonts and OLE controls. A title may have a price associated with it, which is set using the MSN sysop tools.

A title object is a container for sections and pages. A title may also contain supporting objects, including style sheets, content objects, window objects, and resources that need to be installed. A title object has a context menu with common commands, and a property sheet for setting all of its detailed settings.

When a new title is first created, it is populated with the following objects: a resources folder having a default style sheet and a default window, and a blank page (directly under the title object), named "Front Page".

A title object contains pages and search objects organized into sections and subsections, as well as a collection of supporting objects (style sheets, fonts, OLE controls, and so forth). Pages and search objects may be contained directly underneath the title, or they may be organized into a sequence of sections. Windows and style sheets are placement-independent; they may also be stored within sections, or they may be kept in generic folders within the title. Fonts and OLE controls must be kept in the resources folder. The title maintains a sequence ordering of the sections, pages, and search objects, as this is important in determining how the title is displayed. The views supported by the project designer window allow the publisher to re-order these objects.

The title object is the section which represents the entire title. From this root section the hierarchy of sections, search objects, pages and style sheets are added. The title object inherits from the section object and as such, it contains all the properties and attributes of sections.

From a COS perspective, the title object is the root object in an object store. The title has a reference to all first level sections, that is, it is the root section.

Like all COS objects, a smart pointer (CTitleSPtr) object is defined to access the methods of a CTitle object. The CTitle object (like other COS objects) does not have knowledge of the COS which contains it. This information is kept in the smart pointer and all access should be through a CTitleSPtr object.

C. Section

Like the title object, a section object behaves much like a folder. It, along with its contents, can be dragged and dropped. The only visible difference is that its contents has a sequence that can be user-defined.

Sections having Dynamic story controls may set whether the Viewer **202** begins each story on a new page, or runs the stories together. For sections having static story controls, this setting is disabled.

The content gathered into a section is sorted at pressing time before it is displayed. The designer may specify the sort order that will be used at that time. Available sort orders include: Priority, wherein the author or editor may tag each piece of content with a priority, (a number from 1 to 5, where priority 1 is the highest priority), and Date/Time.

The designer may choose a specific page to use when the viewer jumps to a story out of sequence. This allows the system **100** to quickly compose the story without needing to compose all pages in between the current position and the desired story. Without the ability to do this, navigating to specific stories within a section would be painfully slow.

The sequence of pages and search objects in a section help to determine how the section will be pressed. However, the rules are different for static and dynamic sections, as described below.

A static section has all of its content hand-placed on pages. It does not use dynamic story controls to display content. In a static section, the pages are presented to the end-user in the order in which they appear in the section.

A dynamic section uses the Dynamic Story control to display stories within a section. The stories are sorted according to rules specified on the section's property sheet and then are concatenated or linked together. The stories are then filled into the dynamic story controls on each page in the section, in the order in which the pages are arranged in the section. If there are more stories than there are pages, the last page is re-used repeatedly until all content has been pressed.

When in "home delivery" mode, all of the stories are pressed at once; otherwise the system presses pages on demand. In the "on demand" mode, if the user chooses to jump to a story out-of-sequence, the MP system **100** must choose a particular page in the sequence to start displaying that story. Because the MP system **100** has not had an opportunity to press all of the stories before the one that is jumped to, it does not know which page would be used if the user read the stories in sequence. The designer may mark a particular page to be the "out-of-sequence" page, which is then used for these cases. Search objects are activated in the sequence in which they appear in a section. Because of this, the ordering of search objects in a section has a subtle influence on the final ordering of contents. The sorting specified by the section is done as a "stable sort". For example, if two stories, gathered by different search objects, sort to the same place, the one whose search object was listed first in the section will be listed first.

The sequence of pages and the sequence of search objects have no relationship to each other. They may appear interleaved in the section container with no unusual effects.

The section object provides the hierarchical structure of a title. It can be thought of as a folder which can contain other sections (sub-sections), search objects, style sheets, pages, and content. The different objects are managed in separate lists and because of this there are similar but distinct API's to access each of the object types. There is no single homogeneous list containing all the objects that are part of a section.

Sections provide logical breaks in a publication. For example, the different parts of a newspaper can be represented by sections: Front page, Sports, Lifestyles, and so forth. Sections also play an important role in the composing and navigation features of the MPS as follows:

i) Dynamic navigation to sections via information maps

ii) Direct navigation to sections via action controls

iii) The first story of the section is always composed on the first page of a section.

iv) Pages from previous sections do not propagate to subsequent sections. Each section contains the pages which are to be used to compose it's stories.

v) The search objects contained in the section define the content that will be displayed there.

Ordering of objects in a section is important. The title is composed top to bottom, with sections at the top being composed before sections further down. Similarly, pages are used in the order they appear in the section, and dynamic stories appear in the order of the search objects in a section.

A smart pointer (CSectionSPtr) object is defined to access the methods of a CSection object. The CSectionSPtr will instantiate the CSection object from the COS and through it's overloaded arrow operator, allow access to the section methods.

d. Window

A window is a place a page can be displayed. A window may have a fixed size, or the height and width may be

changed at runtime to match the page being displayed in the window. A window object may reside within a section in a title, or it may be contained within the generic resource folder in the title. The designer makes the decision about where the window object is to be contained.

The window object acts as an OLE frame object and client site for an embedded page. The page object (or any compound document server) and the window object interact solely through the OLE Compound Document interfaces. The window object and page object together provide a full in-place capable OLE container. The window object implements the IOleInPlaceUIWindow for its document and the IOleInPlaceFrame interfaces for its frame. It provides the object window (in the Windows sense of the word "window") and it also provides menu bar handling.

By breaking up an OLE in-place container into two parts (window object and page object) using IOleInPlaceUIWindow::SetActiveObject, the page object can be attached and detached from a given window object at run-time. This approach saves time by not having to destroy and re-create a Windows window every time a new page is needed to display data. This is useful within the MP system **100** because it is very common to use two pages (e.g., SectionFront and SectionDetail) to display a particular portion of content.

e. Page

A page is the representation of the layout of the client area of a window. The layout within that page is represented by control objects. A page is a container for controls. In the presently preferred embodiment, a Component Forms development environment for pages operates like Microsoft Visual Basic or Microsoft Publisher in that designers place objects (controls) on a page by selecting and placing them. When opened, a page presents itself in "design mode" in which the user can lay out the page with controls. Properties of the objects are available through property browsers or a context menu.

Pages are contained within sections. Within a section, the pages have a sequence that determines the order in which they are used to press content and the order in which they are displayed when the user browses sequentially.

A page may be used in more than one section; this is accomplished by making a shortcut to the page in another section. A shortcut to a page has a place in the sequence of pages just as the real page does.

A page object is an implementation of an OLE compound document server. The page object can contain any arbitrary OLE control to create any type of Windows application. The page object supports OLE controls fully and is a full compound document server that also supports in-place activation with the IOleClientSite, IOleInPlaceSite, and IAdviseSink interfaces on its site objects. Unlike most OLE implementations, the page object requires a window object to become an OLE container. The window object wraps a page object and displays it in a top-level window or child window. To work properly with controls, the page object also implements an IDispatch interface for ambient properties and an IDispatch interface for control events on its sites, along with a new IControlSite interface that serves as a notification sink for changes in a control's mnemonics.

f. Search

Search objects describe how to collect stories to be pressed into a dynamic story control. Only sections with pages having dynamic story controls take advantage of search objects. A section with one or more dynamic story controls may have zero or more search objects collecting stories. If two search objects collect the same story into a section, then only one instance of that story is actually collected and displayed in outline controls and in the dynamic story controls.

g. Style Sheet

The style sheet object encapsulates a mapping of style-names to formatting instructions. The style names are a fixed set that authors can apply to their stories. In this way, authors do not need to concern themselves with formatting their content; they simply use a standard set of styles, and the formatting is automatically applied at pressing (using the style sheet that was designed into the title). Since outline, static story, and dynamic story controls rely on separate content, they use style sheets as the basis for their formatting.

Opening a Style Sheet shows a list of the styles available. The publisher may select a style and edit its properties by pressing the Modify . . . button. This displays a tabbed property sheet that lists all of the formatting settings.

There are three types of styles: character, paragraph and wrap. Character styles may be applied to a selected set of characters. They include only those settings that can be applied to an arbitrary set. Paragraph styles (displayed in bold in the list) are applied to the entire paragraph. They include extra settings that affect how the entire paragraph is laid out. Wrap styles (displayed in italics) classify wraps as to their functional use, so that they can be fitted to a particular area of the outline, static story, or dynamic story control.

h. Extensibility

The designer may extend the design environment by adding new OLE controls. These controls may support a wide range of functionality. They may provide generic display capabilities (e.g. video, audio, sprites), or they may utilize the MPS Information Map interfaces to provide new means of navigation. OLE controls may also add additional actions to a list of available actions.

i. Content Folder

A content folder is a container of story objects. It may contain story objects, and other folders. When a project is released, the contents of each content folder in that project are copied to their respective source on MSN **122**. Search objects are set to look for content within a specific source. Through the Project window, the publisher may place stories within these content folders so the search objects can find them when the stories are released to the source on MSN **122**.

Content folders are containers for titles and for story objects. They appear as top-level items underneath the Project, and may have further sub-folders. Search objects use Content Folders to identify the scope of their query.

Content objects are represented by a generic document object within the designer. There are two kinds of content objects: Stories and Pictures. Stories represent MPS stories while pictures represent wavelet or metafile pictures. When the New Content command is used to load in a new picture, the picture is automatically converted to a wavelet, unless it is a Windows metafile (in which case it is left unconverted). Pictures which are saved as wavelets also have a property page which allows for controlling the amount of compression that is done on the image prior to sending it to the viewer for display.

Objects may be drag-dropped from the desktop or the file system into a content folder. These objects will then be released to the Data Center **242** when the Release command is selected. When objects are brought into the system **100**, the project stores a path to the original file that was copied in.

J. Resource Folder

Titles may need additional resources, such as fonts and OLE controls. The designer adds these items into a project by drag-dropping (or copy-and-pasting) them into a title. Each title has a Resources folder directly underneath the title object, with subfolders named Fonts and Controls. The designer adds new items to the title by dropping them into those subfolders.

The designer may also place window and style sheet objects within those folders, which are not necessarily associated with any particular section. This allows for managing the supporting resources separately, or for keeping them with the sections that use them.

The MP system **100** assumes that all of the OLE controls and fonts in the Resources folder need to be installed on the customer workstation **182** to run the title.

k. Shortcuts

Just as in the Windows 95 Explorer interface, shortcut objects may be created and used through the project in place of the objects to which they refer. For instance, if the designer wished to use the same page in five different sections, then the designer can create the page in one location and place shortcuts to that page in the other four locations. Since none of the object's properties are duplicated, changing the original will change its use in all the places where shortcuts to it exist. Shortcuts are sequenced exactly as their referenced objects would be sequenced. For example, a section can contain a mixture of pages and shortcuts to pages, and the two could have a single, potentially intermixed, sequence. The "Go to specific page" action takes as a parameter any page or shortcut to a page. Shortcuts must point within a title or content folder. Shortcuts may not point between titles or content folders.

3. Project Editor

The Project Editor **184** provides the user environment and editing facilities for creating and editing MPS projects and titles. The Project Editor **184** limits itself to defining the structure and organization of the title, leaving the page layout and content definition to other parts of the MP system **100**.

The publisher interacts with objects in the title through the Explorer-like UI provided by the project editor. The left pane of the editor contains the title structure elements (e.g., sections) and the right pane contains the objects contained in that section (e.g., search objects, pages, and so forth). Through the project editor, the publisher adds the sections, search objects, style sheets and pages which define the structure of a title. The project editor uses a drag-drop metaphor for moving and copying objects within, and between, titles.

The project editor is the central editing point in design mode, and as such it interacts with the search object, stylesheet, and page editors to configure and set properties on the title objects. In most instances these editors are invoked by double clicking on an object in the project window. The project editor also supports the idea of styles where the properties of an object, say of a search object or section, are based on an existing search object or section.

i) Interfaces

The project editor provides two types of interfaces: 1) standard MFC C++ interfaces to integrate into the framework, and 2) an ITitleEditor interface to support the command architecture. Specifically, this interface is used to add, remove and modify objects in the project editor. Typically, these are invoked through menus or other UI components, but they could also be called via automation.

The following services are required from other subsystems:

Search Object Editor—the ability to invoke the search object editor to edit the properties of a search object;

Designer/Forms 3—the ability to launch the designer to modify a page;

COS—all services are required; directly adds or queries the COS for objects through smart pointers; a smart pointer is a class wrapper which uses an IObjectStore COM interface to make programmatic use of the COS transparent; the COM interface can be used directly;

Style Sheet Editor—the ability to edit the properties of a style sheet by invoking the style sheet editor on a selected style sheet object.

ii) New Object

A "New" menu command in the Project Editor **184** cascades to show a choice of objects that may be created in a selected container. Table 1 lists the items (in order) that appear on the New menu when an object is selected in the left-pane:

TABLE 1

SELECTED OBJECT	NEW MENU CONTAINS
Project	Title, Content Folder
Title	Folder, Section, Window, Page, Search Object, Content, Style Sheet
Section	Section, Window, Page, Search Object, Content, Style Sheet
Content Folder	Folder, Content
Folder (in a Title)	Folder, Window, Page, Content, Style Sheet
Folder (in a Content Folder)	Folder, Content,

4. Page Editor

When a page object is selected in the Project window and the Open command is selected (or the page is double-clicked), the page is opened into a Page Editor window for editing.

The Page Editor is the tool for creating and editing detailed page layouts. The client region of the window represents the page itself, and the rest of the editor window provides tools for laying out controls on that page.

The key elements of the Page Editor are:

i) A Menu Bar with commands for layout and editing

ii) A Toolbar with shortcuts to common commands

iii) A floating palette Toolbox which allows for selecting controls to be dropped on the page

iv) A grid which assists with exact placement of items, and Snap-to behavior which automatically aligns controls with the nearest grid point

v) Undo/Redo support

vi) Nudging support

Double-clicking (i.e. opening) a page object brings up the Page Editor window, showing the layout of that page. Selecting Close from the File menu will close the Page Editor. If any changes have been made that were not yet saved, the user will be asked whether s/he wishes to save those changes before exiting.

The client area of the Page Editor window shows the layout of the page. Each control on the page appears in its appropriate location.

Clicking on a control selects that control; the control then has a sizing border which can be used to resize the control. The control may also be dragged to another location by clicking in the center area of the control (i.e. anywhere within the sizing border). Right-clicking on the control brings up a context menu.

The File→New command cascades to show a list of the available controls. Selecting a control from the cascade menu places an instance of that control on the page in the upper-left quadrant, with height and width one-quarter the height and width of the page, and with the top-left corner at a position one-quarter width and height from the top-left corner of the page.

The Toolbox is a floating window which has a button for each of the controls available to the user to place on the page. This includes all of the standard controls, plus any additional controls that have been dropped into the title. The currently-selected control is depressed, and only one button is depressed at any given time. The toolbox also has an "Arrow" button, which is depressed when no control has been selected and indicates that the cursor can be used to select and reposition controls on the page.

The user adds a control to a page by pressing on the control's button, then dragging the rectangular region on the page where the control is to be placed. The control is created when the mouse-button is lifted at the end of the drag.

5. Style Sheet Editor

The use of style sheets in MPS controls provides a way for the designer to set text formatting properties on a per control basis by creating and assigning a different style sheet for each control or a set of controls. Style sheets are created using the Style Sheet Editor **187** (FIG. 2) provided with the Project Editor **184**. The Style Sheet Editor **187** allows the designer to customize style property values to override the default definitions for styles provided with each title.

The designer uses the Project Editor **184** to insert/create a new style sheet in the Title. Initially, the style sheet is empty and simply uses the style definitions in the default style sheet. The designer can then invoke the is Style Sheet Editor **187** to define properties for the style sheet and change style property values. The set of styles supported by style sheets is predetermined. The designer cannot create new styles but can only modify the default property values for existing styles.

When invoked, the Style Sheet Editor **187** displays a dialog listing the (predefined) style names and fields for editing their property values. These are retrieved from the default style sheet created by the title. Initially, these fields contain the default values defined in the default style sheet. The designer selects the style name of the style to customize and sets the desired new property values. When the properties are set as desired, the designer causes the Style Sheet Editor **187** to create a new style object with the new property values and add it to the style sheet. The new style object now serves to override the default. The designer continues defining new styles in this manner, as desired. When complete, the designer dismisses the Style Sheet Editor **187** and can proceed to assigning the new style sheet to MPS text controls.

6. Search Object (Magnet) Editor

The Search Object Editor **189** (FIG. 2) is a modified version of the customer Find Dialog **510** (FIG. 9). Since the Search Object Editor **189** is to be used by designers for title construction, there are a few differences:

The Search Object Editor **189** is a modal dialog that behaves as the property sheet of a Search Object in the title designer. In the presently preferred embodiment, the Search Object Editor does not allow the designer to "Find Now" and test the query. After the criteria has been entered, the dialog is either committed with OK or dismissed with Cancel. In another embodiment of the system **100**, the designer can select "Find Now" and test the query.

The In: checkboxes directly mirror the fields of the MPS Document Editor Summary Info dialog to give the designer more precise control than the customer when retrieving stories.

In comparison to the Look In: field that appears in the Find Dialog and denotes what finished title(s) to search, the Source: field specifies the repository of stories to search for stories to be flowed into a title. These sources are accessed via the More . . . option at the bottom of the dropdown that launches a tree view of all content sources on the MSN **122**. These sources are only visible to title designers and do not appear to the general MSN public.

The search may be limited to retrieve no more than a certain number of stories to prevent a section from running too long. The designer simply specifies a maximum number in a provided spin control.

7. Controls

This section specifies only the preferred base controls included with the MP system **100**. Other embodiments of the system include controls for animation, database access, electronic commerce, and so forth. The following is a list of MPS specific controls included in the base MP system **100**: Caption, Caption button, Picture, Picture button, Outline, (Static) Story, Dynamic story, Shortcut, and Audio. Each of these controls is further described below.

a. Characteristics

A property sheet having pages is associated with each control. Most of the property sheet pages are used in more than one control. Note that no control has all of the property sheet pages; each control only has a small subset of the pages on its property sheet. Pages which are not applicable to a control do not appear as tabs on that control's property sheet. The property sheet pages utilized by the MP system **100** include:

- General—The General page lists general properties related to size and position.
- Border Page—The Border page allows for setting the style and color of the rectangular frame around the control.
- Action Page—(described below)
- Text Page—The Text page allows for setting the values associated with displaying a text value.
- Appearance Page—The Appearance page is used for setting text-display properties for richer displays than a simple caption (e.g., the story control).
- Story page—The Story page allows the designer to choose the story object that is displayed in a story control.
- Where to Look Page—The Where to Look page is used by information maps to decide which part of the title should be used to display information.
- Bevel Page—The Bevel page is used for setting the bevel attributes of "button" action controls.
- Picture Page—The Picture page is used for choosing a single picture object to display in a control.
- Pictures Page—The Pictures page allows the user to set more than one picture, for example a button control which needs both an up and down picture.

Both pictures must use the same placement, rendering, background and transparency settings.

- What to Show Page—The What to Show page is used by the Table of Contents control to choose specific attributes of content to display.
- Font Page—The Font page allows for choosing a font and style for those controls that have a single, consistent font throughout the control.

- Shortcut Page—The Shortcut page is used for defining a shortcut to a story or an MSN object.
- Color Page—The Color page allows for setting all of the color properties for each control.
- Audio Page—This page is used by the Audio control for setting audio playback functionality.

i) Action Property Page

The Action property page is now further described. Controls such as caption buttons or picture buttons enable customers to invoke actions. A list of actions, provided in Table 2, is extensible by adding new controls which export additional commands. The action property page enables designers to associated these actions with events that can occur on the control. For example, in response to a mouse down event, the designer can associate the action to go forward one page. The standard actions include:

TABLE 2

Action	Description	Parameters
OpenTitleDlg	Brings up Open Title dialog	
SaveAsDlg	Brings up Save As dialog	
UpdateNow	Causes immediate update of title	
PageSetupDlg	Bring up Page Setup dialog	
PrintDlg	Brings up Print dialog	
Exit	Quits the title	
EditCopy	Copies selection to clipboard	
GoBack	Jumps back to last page accessed	
AddShortcut	Adds a new shortcut	
ShowShortcuts	Brings up the Shortcuts folder	
FindDlg	Brings up Find dialog	
PrevPage	Jumps to previous page in section	
NextPage	Jumps to next page in section	
FirstPage	Jumps to first page in title	
HistoryDlg	Brings up History dialog	
InterestsDlg	Brings up Interests dialog	
ScheduleDlg	Brings up Schedule dialog	
PreferencesDlg	Brings up Preferences dialog	
GoToPage	Jumps to specific page	page to jump to
PrevSection	Jumps to first page in previous section	
NextSection	Jumps to first page in next section	
GoToSection	Jumps to first page in specific section	section to jump to
Run	Runs a program or opens a file	file to run
CloseFrame	Closes the current frame	

b. Caption

The caption control is used to display simple textual information. The designer has control of the choice of text, colors, font, size and placement of the caption. The other major use of the caption control is to automatically display information about the title. For example, a caption control can be used to display the current section. This is useful if the page containing the caption is used in many different places.

c. Caption Button

The caption button control is closely related to the caption control. In addition to all the properties of the standard caption control, including the predefined automatic title information options, the caption button has an action page and bevel page.

At run-time, the caption button displays a text caption within a button frame. Clicking on the button will cause a Click event to occur, which will then fire the associated action.

At design time, the caption button displays itself as it would appear in its "up" position. When the control is selected, it has a sizing border.

d. (Static) Story

The (static) story control automatically composes a layout for a stream of textual and graphical information. The layout describes the positions of text and graphics that are presented within the area the story control covers on a form.

The story control is closely related to the dynamic story control. The two major differences are:

- The dynamic story control displays a sequence of stories found in a section of title, whereas the story control stores the information internally.
- The story control can not display information in page mode. Dynamic story controls can display information that spans multiple pages. Note however, that the story control is scrollable, in which case this restriction is less important.

i) Performance

The preferred story control takes no more than two seconds to place text and graphics on a typical screen oriented publication. This is to ensure that the story control can provide layouts at rates comparable with **9600**-baud modem communications speeds. In other words, a typical short text story—the size that would fill the first page of a typical publication—can be delivered to the customer's workstation **182** in approximately two seconds. This would mean that a 30 page publication could be "paginated" in under a minute. Note that the process of creating a layout may occur in the background, so customers may not experience delays when navigating throughout a publication. Once the layout on a page is finished, the time required to repaint a page (excluding graphics) should be small, <250 milliseconds, in order to compare favorably with paper-based publications.

ii) Elements of a Layout

The story control operates on three kinds of elements:

- Tagged text. This is text with logical descriptors, such as headlines, that is styled and placed into a story control.
- Inlines. An inline is a graphic that is associated with a particular point within the text. For layout purposes an inline is essentially treated as a character for layout purposes.
- Wraps. A wrap is a graphic that may be placed somewhere within the control. While a wrap may be associated with a particular point in the text, it will not typically be placed at that point, but instead will "float" to an appropriate point within the presentation.

iii) Types of Graphics

The story control can support any OLE object or MPS custom control. For purposes of composition, the preferred story control treats all graphic as simple rectangles. In another embodiment, other shapes for the graphics are supported. In general, the story control does not know what is contained within a graphic. Consequently the story control considers objects that require interaction with the customer, such as shortcuts, to be simple graphics. The story control ensures graphics are drawn when appropriate and that customer generated events are delivered to graphic objects, such as links represented by a picture, that require the information.

iv) Presentation Options

The story control supports vertical, single-column scrolling. The story control does not support page mode, i.e., it does not display information that spans multiple pages.

v) Story Control Properties

Story controls have the following properties: number of columns, margins, a style sheet (the style sheet maps the

logical tagged source text into rendering information), and a source specification (determines the source of tagged text for the story frame).

vi) Run-Time Mode

During run-time the story control provides distribution of customer events to embedded MPS OLE controls, support for scrolling, and rendering of text and graphics.

vii) Design Mode

During design time, the story control provides the designer with the ability to set properties and edit the content of the control. To edit the content of the control, the designer must edit the story object that the control points to. The designer can access the properties of the control by selecting "Properties . . ." from the context menu of the control.

e. Dynamic Story

A dynamic story control is a story control that has its content dynamically gathered, rather than statically placed at design-time. Each page in a dynamic section may contain one or more dynamic story controls. When the title is pressed, the content is gathered and flowed into the dynamic stories on the pages in that section, re-using the last page until all content has been pressed.

The run-time behavior of a dynamic story control is similar to that of a (static) story control except that when a dynamic story control does not scroll, the stream of data is flowed into the next dynamic story control on the page or to the next page.

During design mode, the designer can set properties of a dynamic story control and can specify the order that the dynamic story controls will display themselves on the page. Properties are accessed via the properties menu item on the context menu.

The dynamic story control is the control that is used to format, layout, and display the story text of the results of a search object on a given page. This control provides a fairly full set of text layout, formatting and display capabilities including text insertion, deletion, replacing; character and paragraph formatting; page setup; line layout; tabs; text wrapping around intrusions (tight and rectangular wrap); in-line graphics; backgrounds; and creation and hit detection of hot links.

The dynamic story control also provides the capability for read-only type user interactions such as mouse-clicking, mouse and keyboard selection, selecting and activating hot links, and copying to the clipboard during view mode.

The main distinguishing characteristic of this control vis-a-vis the (static) story control is that the contents of this control are gathered dynamically via search objects and inserted into the control programmatically rather than created statically at design time by the designer. When the designer creates one of these controls he/she will not author the contents of the text directly. That is, there is no editing capability for this control in design mode.

i) Interfaces

The dynamic story control interfaces with the Viewer, link manager and title manager.

The dynamic story control interacts with the Viewer **202** to accomplish page setup and composition. The dynamic story control gets a parse tree node from the Viewer **202** where composition should begin. After composing, it will notify the Viewer of the last character that was consumed during the composition. Also, when a link is clicked on it will pass the link to the link manager for link resolution (i.e. to traverse the link and move the user to the target of that link).

The dynamic story control registers links with the link manager as they are created. Also, the dynamic story control interacts with the link manager to have the link updated (visually on the screen) after it has been resolved at least once. This indicates to the user that the link has already been resolved at least once and therefore the target of that link already exists locally in the COS.

The dynamic story control interfaces with the title manager to convert style tags to style objects which can then be used to apply the appropriate formatting to the text.

f. Picture

The Picture control displays a bitmap or metafile. It can display the picture all at once, or it can progressively render the picture. The Picture control does not embed the picture within the instance of the control, but rather, it points to a separate Picture object within the title.

The run-time mode displays the picture as configured in the Property Sheet at design-time.

The design-mode of the control is identical to the run-time mode, except that when selected, the control has a sizing border, and the user may set the picture by selecting the control and (from the property sheet) browsing within the title to select a desired picture object.

g. Picture Button

The picture button control provides the designer with an up and down picture and animated button wrapper around the two pictures. This control supports the following events: click, right click, mouse down and mouse up.

The Run-time mode displays the "up" picture until the user clicks down on the control, at which point it then displays the "down" picture. The Design mode displays just the "up' picture, with a sizing border when the control is selected.

h. Outline

The outline control is used to display information about the title and the stories contained within the title. During run-time mode, this control displays sections (if chosen) and tags (those selected in the property sheet) and displays them according to the style sheet selected. Clicking on any item will navigate to that item in the title. During design mode, this control shows sample displays for each of the types of content chosen in the property sheet, using the selected style sheet.

i. Shortcut

The Shortcut control is used for defining a shortcut to a specific story or a Microsoft Network object (title, chat or BBS session, and so forth).

In run-time mode, the control is transparent, except for the standard shortcut icon in the bottom-left corner. When the mouse-cursor is over a shortcut control, the cursor changes to a hand; this is behavior is the same as for the button and outline controls.

The design-time behavior is very similar to the run-time behavior, except that when selected, the control has a sizing border that allows the control to be moved and resized. It also has a standard context menu and property sheet.

j. Audio

The audio control allows for playing an audio object when a page is active. In run-time mode, the control is transparent. The design-time behavior is very similar to the run-time behavior, except that when selected, the control has a sizing border that allows the control to be moved and resized, and the word "Audio" is repeated throughout the control region. It also has a standard context menu and property sheet.

The Audio control adds the following actions to the list available to button controls:

- Play—begins playing from the beginning of the clip
- Stop—stops playing, sets current position back to beginning of clip

- Pause—stops playing, maintains current position
- Resume—begins playing from current position None of these actions take any parameters.

k. Wraps (Intrusions)

The process of getting a wrap placed correctly in a dynamic story control, outline control, or story control has three steps:

1) The author places the picture in a content stream, and marks it as a wrap.
2) The title designer (not the author) decides where the wraps should be placed within the control.
3) When the customer runs the title, the control code sees the wrap and places it according to the designer's settings.

Of course, the difficulty in this is to identify the means by which authors mark a wrap as such without actually placing it, and by which designers set rules for placing wraps without prior knowledge of what the wraps are. This problem is solved by using styles. The user has eight wrap styles available within the MPS Document Editor authoring environment.

C. Architectural Structures

This section will describe the structures utilized during the title creation and title publishing processes.

1. COS

Referring now to FIGS. **11***a*, **11***b* and **11***c*, a structural diagram of the COS component of the MP system **100** will be described. As previously shown in FIG. **10**, wherein a Publisher COS **544**, a Server COS **540** and a Viewer COS **548** where described, the COS is an essential component of the presently preferred MP system **100**.

The COS is used for persistent object storage by the MP system **100**. Several different abstractions are exposed by the COS: a simple stream for arbitrary storage, a discreet and user-transparent object storage for MFC CObject-derived object classes, and object properties. A simple stream can be used to contain an array of bytes of data for whatever purpose. A CObject-derived object is managed persistently by the COS once created and added to the system. A property is a named typed value and can be associated with discreet objects and streams. The COS interface is thus divided among COS compound document file creation/opening/closing, simple COS stream access, COS object persistence, and stream or object property access.

The following references are useful for understanding the COS subsystem.

OLE 2 Programmer's Reference Volume One, Microsoft Corporation, 1994, Chapter 9—Persistent Storage Interfaces and Functions;

Class Library User's Guide For the MFC Class Library, Microsoft Corporation, 1993, Chapter 14—Files and Serialization;

OLE 2 Classes For the MFC Class Library, Microsoft Corporation, 1993, class COleStreamFile

At the lowest level, the persistent object storage strategy is to leverage OLE 2.0 capabilities for compound document access, which includes an internal IStorage hierarchy and IStreams for data content. Objects and property information describing objects are stored into discrete IStreams, which are organized within a hierarchy of IStorages.

Referring to FIG. **11***b*, an object **580** is exemplary of an IStorage, while IStreams for a typical object include an object data stream **582**, an object properties stream **584**, and a handle table stream **586**. The object data stream **582** is required for the object, while the object properties stream **584**, and the handle table stream **586** are optional. The property stream **584** is not needed if properties are not supported by the object type. The handle table stream is not required if the object doesn't reference anything.

The COS is the basic component which mediates persistent storage of MPS objects into an OLE compound document file. It is the go-between for both MPS applications and tools and also mediates access to a COS compound document on behalf of the Object Broker **550** (FIG. **10**).

The Object Broker **550** provides for peer-to-peer communication for distributed object needs of the MP system **100**. An Object Broker **550** resides on any machine that connects into the MP system **100**—either as publisher, server site, or customer (client).

The Object Broker **550** interacts with the COS for compound document access and fundamentally transacts with MPS client software applications via OLE COM RPC. In the case of operating in the context of Microsoft Network Online Service (MOS), the Object Broker **550** also makes use of MOS MPC (remote procedure call mechanism). MPC is used for Object Broker-to-Object Broker, peer-to-peer interactions. OLE COM RPC is used for both client-to-Object Broker server and the server-to-server scenario where network connections and protocols supporting COM RPC permit.

The COS component library is intended to be statically linked into any piece of software using its services. It will support process internal multi-threading concurrency. COS services are accessed through a C++ class interface. The Object Broker **550** is implemented as a process server using process internal multi-threading concurrency to service potentially multiple clients.

The central abstraction of the COS is that of a COS-locally scoped handle to an object moniker, which in turn associates a persistent object. The object moniker is key to the functionality of the system. An object moniker is a special object in itself (is an object used in the implementation of the COS). It dual encapsulates information on either the local cached location of an object, and/or contains the GUID identity of the object which can be used in its retrieval from a remote location. The object moniker is also used for reference count access tracking, object dirty status, COS object type information, and a few other items. The object moniker must be used by the COS when actually dereferencing a COS object so that proper reference count tracking, dirty status, and so forth, can be kept properly synchronized.

A GUID is assigned to an object to uniquely distinguish that object. A GUID is generated by an OLE 2.0 API, known as CoCreateGUID, using both a time identifier and a machine (computer) identifier. This guarantees that any two GUIDs produced on the same machine are unique because they are produced at different times, and that any two GUIDs produced at the same time are unique because they are produced on different machines. The presently preferred GUID is a 128 bit or 16 byte number.

Referring to FIG. **11***c*, an object moniker represented as a moniker table record **630** in a moniker table, such as moniker table **600** shown in FIG. **11***a*, will now be described. A moniker table is persistent, that is, it is a stream at the root level of a COS. Structurally, the moniker table is preferably implemented as a sparse array, wherein if the array doesn't need a slot in the table, the memory is released. The moniker table record **630** includes information that uniquely defines an object stored in the COS. The fields of the record **630** include a GUID field **632**, a publish date field **634**, a handle or path to the storage **636**, a set of flags **638**, and a persistence reference count **640**. The path **636**, also

called an object handle or CDPOHandle, is a DWORD (32 bits) that provides a short path or name of the storage. The flags **638** indicate the existence of artifacts, that is, whether the data stream **582**, the properties stream **584** and the handle table stream **586** exist in the COS for the current object. The persistence reference count **640** is only valid within a COS and is used for "garbage collection". When an object is no longer used, signified by the persistence reference count **640** equal to zero, garbage collection removes the object. When the object is removed, a tombstone is created in the moniker table to indicate that the object did exist in the COS at some time in the past. The tombstone is a moniker with a flag which indicates that the object of that GUID is extinct. The object storage is purged but the moniker remains.

The object moniker is also instrumental in the MPS scheme for local object caching. The GUID identity of an object encapsulated in an object moniker is used to seek and identify the appropriate object remotely from the machine it is requested for. A user may have downloaded a title COS from an on-line MSN server, such as MPS server **246** (FIG. **3**). This title is stripped of all objects leaving only a moniker to the root title object. Attempting to view this title and thus attempting to access its objects forces absent objects to be remotely retrieved via the Object Broker. The objects become stored locally (in the Viewer COS **548**) with the Object Broker **550** on the machine initiating the request. Viewing a title having locally cached objects then subsequently offers much higher performance. Yet, the objects comprising this title can be tracked and updated over time, such that the title remains fresh with the current edition mastered by a publisher. When a title is on a customer's machine, it is most appropriate to think of the title's objects as being remotely served and mastered, but locally cached for optimal user performance of perusal.

The Object Broker component resides upon all nodes in the MP system **100** comprising publishers, title servers, and client end users. The Object Broker primarily fields requests for objects specified by GUID. If one Object Broker server cannot fulfill an object request, it attempts to relay the request to server sites that it knows about. The object, when and if found, can later be propagated to its final destination in a store-and-forward manner, or the requested object can be transmitted between the point of its discovery to the point of its request without store-and-forward copies of it being retained at intermediate server sites. This is governed by system configuration settings for each Object Broker site and the administration policy of that site.

The nature of the ultimate repository of an object can be rather flexible under this scheme. The key item is the object moniker and the GUID it encapsulates. The object moniker could be extended so as to provide for arbitrary mechanisms by which to synthesize the actual object at object request occurrences. The object may reside in a COS IStream. The object could reside in a file system file external to the title COS file. Or the object could reside in some other foreign object data base and thus require some protocol interaction in conjunction to that data base to retrieve the object (and perhaps later commit modifications to the object). This scheme of local caching effectively shields MPS client applications, tools, custom OLE controls, and so forth from having to have specific knowledge as to how the Object Broker go-between process actual does its task of retrieving the desired object and locally caching it.

It is important to note that when an object becomes locally cached, it resides in a COS IStream—within a special COS belonging to the machine local Object Broker (see FIG. **16**). This COS belonging to the machine local Object Broker may be hidden to protect the customer from inadvertently deleting or moving retrieved object data that is pertinent to the user's downloaded title(s). The Object Broker is responsible for periodically expunging stale objects from this special COS so as to conserve user hard disk space.

2. SuperCOS

Referring again to FIG. **11***a*, a discussion of an exemplary superCOS **570** now follows. The preferred superCOS is a COS that references one or more COSes, which are known as subCOSes. In the exemplary superCOS **570**, there is a root portion **572** and two subCOSes, a title COS **590** and a content folder COS **592**. In the root portion **572** of the superCOS **570**, there is a storage for types. As shown in FIG. **11***a*, there is a project type IStorage **576** off the root of the superCOS **570** which references one or more storages for each object of the type, which is project in this instance. The project object **578** has the three IStreams, such as the streams **582**, **584**, and **586**, as previously described in conjunction with FIG. **11***b*. The project object **578** keeps track of the subCOSes **590** and **592**. That is, the object data stream **579** of the project object **578** references each subCOS created by the publisher for the current project. The root portion **572** also includes a moniker table **574**. Since the root portion **572** preferably only includes a project object **578**, the moniker table **574** only has one record and is a persistent stream off the root of the superCOS **570**.

The relationship between a title and a COS is one to one, that is, there is one COS per title. This relationship follows from the concept that a COS is a distinct set of objects that reference each other. In the present embodiment, a title does not reference another title within a superCOS, but this capability is provided in another embodiment.

The title COS **590** includes a moniker table **600** and one or more type storages **594**. The type storage could be for a title, section, page, style sheet or other MPS objects. In this example, type storage **594** references an object **596** and an object **598**. A moniker table record is created for each object in the COS **590**. In general, a COS has "n" types, and within the type storage, there may be "m" objects.

The content folder COS **592** includes a moniker table **601** and a type storage **602**, which in the preferred embodiment is of a folder type. The storage **602** references other containers, such as a folder (also called a subfolder) **604**, and content **606**. The use of subfolders is optional, but they serve to help organize objects. The content folder COS **592** provides a means of publishing fresh content. The publisher could create a title stored in the title COS **590** for which the layout changes very infrequently, if at all. Then, periodically (e.g., daily, weekly, monthly), the publisher can publish new content (e.g., stories, images, bitmaps, and so forth) in the content folder **592**.

Referring again to FIG. **11***c*, a GUID map **620** will be described. When a COS is opened, the MPS reads out the moniker table and creates the GUID map **620**. The GUID map **620** is created in the RAM of the computer where the COS exists in the form of a hash table structure. The map **620** is created by reading each moniker table record to extract the GUID, such as GUIDx **622**, and storing the GUID and a pointer **624** to the corresponding moniker table record **630** in the GUID map **620**. Each GUID stored in the GUID map **620** has a pointer to one moniker table record. Thus, given a GUID, the system can access the moniker which provides the path to get the corresponding object and its artifacts.

3. Title COS

Referring to FIG. **12**, an exemplary title COS **591** will be described. The title COS **591** is similar to the title COS **590**

of FIG. 11*a*, but is enlarged to show a plurality of different type storages and referenced objects. The title COS **591** includes a moniker table **600** with a record **680–690** for each object in the COS **591**. The type storages in COS **591** include a style sheet storage **650** having a default style sheet object **652** and a second style sheet object **654**, a title storage **660** having a title object **662**, a section storage **664** having a section object **666**, a optional page wrapper storage **668** having a page wrapper object **670**, and a page storage **672** having a page object **674**. Each of the COS objects (title, section, folder, content folder, root content folder, proxy table) nominally has the three artifacts for data, properties, and handle table as previously described in conjunction with FIG. 11*b*. The exceptions are that window, page, content, search, and style sheet objects normally do not have a handle table artifact, e.g., default style sheet **652** has a data stream **656** and property stream **658**. These objects are leaf objects that do not reference other objects. Another exception is that a certain type of page object, the virtual page object **674**, normally only has a data artifact **673**.

In the presently preferred embodiment, the optional page wrapper object **670**, which has a data artifact **669** and a properties artifact **671**, is included as a design convenience. In another embodiment of the system **100**, the page wrapper storage **668** and page wrapper object **670** are not utilized.

When a new title is created, the title references a default window object, having standard window properties, and a default style sheet object. The title designer can choose to modify the default object, and thus create, a new window object to exhibit the behavior desired for the window. A similar process can be done for the style sheet object.

A handle table **663** for the title object **662** has information for all the sections in the title COS **591** (in this example, there is only one section shown). The title object **662**, therefore, persistently has knowledge that the title object **662** references the section object **666**. If a new section is saved in the COS **591**, the handle object **663** is updated with information so as to reference the new section object.

OLE Controls are not objects in the MP system **100**. OLE controls are stored as data for the page in the data stream. Each control has a Class ID and an OLE ID that is not a GUID. The properties of some controls have a GUID to a style sheet.

At the viewer COS **548** (FIG. **10**), when a title is accessed for the first time, the title is preferably pulled over from the server COS **548**. Alternatively, the title could be accessed from the CD-ROM **124**. When the title is accessed for the first time, the Viewer **202** (FIG. **2**) creates a moniker record **684** for the title object in the moniker table **600**, gets the data stream and gets the handle stream. The Viewer **202** incrementally brings over objects into the viewer COS **548**. Starting with the title object, the handle stream **663** has the GUIDs for the other objects the title references. The viewer **202** initially puts the GUID for each object, for example, section object **666**, in a moniker table record for that object (e.g., record **686** for section object **666**). Later, when the object is invoked or called, the Viewer **202** gets the rest of the moniker record information and stores it persistently in the moniker table **600** along with the object artifacts.

4. MSN Server COS

A server COS, such as COS **544** in FIG. **10**, is similar to a superCOS **570**, but does not utilize a project object **578** to reference and keep track of the subCOSes. At a network server, such as MPS server **246**, the server COS does not know what objects will be published to it and which subCOS the objects are associated with.

The server COS **544** contains titles in entirety, since they are published there and the server acts as the master repository. In contrast, a customer superCOS, such as at customer workstation **182** (FIG. **2**) is a cache for those objects which have been transmitted to the customer, and so may only contain a subset of the objects for a title.

D. Operation

An operational flow of publication and viewing will be described in this section.

1. Designer Process Flow

Referring now to FIG. **13**, a title creation process **710** will be described. This process **710** is preferably performed at the publisher workstation **180** (FIG. **2**). This process corresponds with states **324–328** of the top level process **320** (FIG. **5**).

Beginning at a start state **712**, the designer environment is evoked by the publisher **102** (FIG. **1**). In the presently preferred embodiment, the publisher clicks on a Designer icon on the Windows 95 desktop or selects Designer through the menu system of Windows 95. In either case, the bbdesign.exe program, previously described, is initiated. Moving to a state **714**, the process **710** creates a new object **716**, e.g., a project, a title a content folder and so forth. This state is part of the viewdll.dll.

Proceeding to a decision state **718**, the process **710** determines if the object is a project object. If so, the process **710** continues at a state **720** and creates a new COS file to represent the project. This state is part of the cos.dll. If decision state **718** determines that the object is not a project object, the process **710** moves to a decision state **722** to determine if the object is a title or root content folder. If **80**, the process **710** advances to a state **724** and creates a new subCOS, such as subCOS **590** or **592** (FIG. 11*a*) in the project file. This state is part of the cos.dll.

If decision state **722** determines that the object is not a title or root content folder, the process **710** moves to a state **726** and associates the object with the parent container object, e.g., a section is associated with the title object. The object handle (CDPOHandle) of the object, which is a 32-bit local path, is added to the handle table stream of the parent container object. This state is part of the viewdll.dll. Proceeding to a state **728** to mark the properties of the parent object as "dirty". "Dirty" signifies that the memory image of the parent object's properties has been modified. This state is part of the cos.dll.

At the completion of state **720**, **724** or **728**, the process **710** continues at a decision state **730** to determine if the user desires to save the project. If so, the process **710** proceeds to state **732** and commits (i.e., stores) the new and dirty objects to the appropriate COS. This state is part of the cos.dll. At the completion of the commit state **732**, or if it was determined at decision state **730** that the project is not to be saved, the process continues at a decision state **734** and determines if the user desires to create another object. If so, the process **710** loops back to state **714** to continue the title creation process. If not, process **710** moves to end state **736**, wherein the designer environment is shut down.

2. Release Process Flow

a. Client

Referring now to FIG. **14**, a title publishing process **750** will be described. This process **750** is preferably performed at the publisher workstation **180** (FIG. **2**) and corresponds with state **332** of the top level process **320** (FIG. **5**).

Beginning at a start state **752**, the process **750** moves to state **754** and initiates the publish title process. This state is part of the bbdesign.exe. Moving to a state **756**, process **750** compress the COS objects. This state is part of the cos.dll. The process **750** continues to a decision state **758** to determine whether the publisher desires to publish to the remote

server. The designer is given the option to either release the title to the remote server **246** at the MSN Data Center **242** (FIG. **3**) or to create a local COS file which contains all the objects for that title. This is useful for local testing or for releasing a stand-alone title on a floppy or CD-ROM.

If the publisher desires to only release the title locally, the process **750** proceeds to a state **760** and create a local COS file for the current title on the publisher's workstation. This state is part of the cos.dll. At the completion of state **760**, process **750** finishes the title publishing process at an end state **778**.

If the publisher desires to release the title to the remote server **246**, as determined at decision state **758**, the process **750** proceeds to a decision state **762** and determines if there is a previously published timestamp for the COS. If so, the process **750** moves to a state **764** and creates a temporary COS file which contains only those objects which are new, modified or deleted since the last published timestamp. This state is part of the cos.dll. If there is no previously published timestamp, as determined at decision state **762**, process **750** proceeds to a state **766** and creates a temporary COS file for the entire title. This state is part of the cos.dll.

At the completion of either state **764** or state **766**, the process continues at a state **768** and transmits the temporary COS file to the data center **242** (FIG. **3**). This state is part of the mwrap.dll, previously described. Proceeding to a decision state **770**, the process **750** determines if the data center has confirmed the publish operation. If not, the process **750** moves to a state **772** and notifies the publisher of a failure to publish. This state is part of the bbdesign.exe. After the publisher is notified, process **750** finishes the title publishing process at the end state **778**.

If the data center has confirmed the publish operation, as determined at the decision state **770**, the process **750** advances to a state **774** and timestamps the current publish date in the root moniker of the title. This state is part of the cos.dll. Then, at a state **776**, the process **750** notifies the publisher of a successful publish operation. This state is part of the bbdesign.exe. After the publisher is notified, process **750** finishes the title publishing process at the end state **778**.

b. Server

Referring now to FIG. **15**, a title publishing process **790** will be described. This process **750** is preferably performed at the MSN data center **242** (FIG. **3**) and corresponds with state **332** of the top level process **320** (FIG. **5**).

Beginning at a start state **792**, the process **790** moves to state **794** and receives the COS file transmitted from the publisher workstation **180** (as described in conjunction with FIG. **14**). This state is part of the bbobsvc.dll. Moving to a state **796**, the process **790** notifies the Arbiter **252** (FIG. **3**) at the data center **242** of the received COS file. This state is part of the bbobsvc.dll. Advancing to a state **798**, the Arbiter **252** notifies each MPS server replicant (such as **246***a*, **246***b*, **246***c*, as shown in FIG. **3**) of the received COS file. This state is part of the MSN arbiter service.

Proceeding to a state **800**, each MPS server replicant **246** updates its existing superCOS with objects from the received COS file. This state is part of the bbobsvc.dll. To facilitate this update, process **790** moves to a decision state **802** to determine if the COS has been previously published to the data center **242**. If not, process **790** proceeds to a state **804** and adds the entire received COS file as a new subCOS in the superCOS of each MPS server **246** at the data center **242**. This state is part of the cos.dll. A GUID lookup map and subCOS directory in the root portion of the superCOS is updated to reference the new subCOS. Every superCOS has a GUID lookup map and a subCOS directory to identify each object within the superCOS and which subCOS a particular object resides within. When a new subCOS is added to a superCOS, the subCOS directory of the superCOS is updated with the new subCOS. In addition, every object within the new subCOS is entered into the GUID lookup map with a reference to the new subCOS. At the completion of state **804**, the title publishing process **790** (at the data center) finishes at an end state **806**.

Returning to the decision state **802**, if the COS has been previously published to the data center **242**, process **790** advances to a state **808** and extracts a copy of the previously published subCOS from the superCOS of each MPS server **246** at the data center **242**. This state is part of the cos.dll. Continuing at a state **810**, the process **790** updates each subCOS copy with the new, modified, and deleted objects of the received COS file. This state is part of the cos.dll. Moving to a state **812**, the process **790** replaces the previously published subCOS in the superCOS of each MPS server **246** with the newly updated subCOS. This state is part of the cos.dll. At the completion of state **812**, the title publishing process **790** (at the data center) finishes at the end state **806**.

3. Object Broker Service

Referring now to FIG. **16**, the Object Broker and its interaction with and fundamental use of the COS in the context of the MP system **100** will be described. The COS **544** (which may be a superCOS if more than one COS is present) and the Object Broker **546** are shown at the publisher location **102**. The COS **548** (which may be a superCOS if more than one COS is present) and the Object Broker **550** are shown at the customer location **160**. The superCOS **540** (a plurality of title COSes **854** and title COS **840** are shown in this example) and the Object Broker **542** are shown at the server **246**. A title **840** at the publisher COS **544** contains all the objects (signified by the bold circles for truly present objects) while the Object Broker **546** does not contain any objects (signified by the lighter circles for objects not truly present) for that title. On the server **246**, all the objects are deposited into the Object Broker **542**, and only a skeleton title **840** remains (title with all the object removed having just a root moniker). At the customer site **160**, the skeleton title **840** is used to start viewing a title, and the Object Broker **550** will contain those object necessary to view the title.

As previously described, the COS can be considered an OLE compound document file utilized for persistent object storage by the MP system **100**. COS manager is the code associated with accessing a COS file and is an implementation of the IObjectStore interface, residing in the COS DLL. The COS manager is the intermediary between a client application and this object storage compound document. It manages the insertion and retrieval of persistent objects to the COS compound document, at the lowest level.

The COS manager can field object retrieval requests locally from its COS context. In the event the requested object is not physically present, the COS manager can in turn field the request to the host machine Object Broker (the local Object Broker at the customer workstation **182**, for example). The local Object Broker may have the requested object cached on the local host machine, but if not, it will seek to obtain the requested object from a connected remote server **246** at the data center **242** (FIG. **3**). But this chain of events may continue in the event that the remote server **246** does not have the requested object cached either, in which case it will in turn propagate the request to servers that it is connected to. If eventually found, the requested object may be either directly forwarded to the original requester

machine's Object Broker, or it may be store-and-forward propagated through the chain of connected server sites. At each server point a MPS Object Broker is the mediator for the object request. This is the Object Broker's primary role. The COS Object Broker interface is implemented in the convention of an OLE COM custom class and is called IObjectBroker.

Secondarily, the Object Broker provides a special publishing operation **842** on behalf of MPS publishing sites, such as publisher **102**. A fully produced publisher's COS document, known as a title **840**, is propagated to server sites **242** via use of this publishing operation **842**. During this process, the title COS **840** is stripped of its objects to where only object moniker tables remain in the title COS **840** at the server superCOS **540**—and perhaps optionally a select group of exempt objects that continue to reside in the COS **840**. Most of the actual objects of this title COS **840** are propagated to and update the server's Object Broker local object cache **832**. The Object Broker local object cache **832** is a COS file made hidden via the file system. As previously described, an object moniker is a special object proxy that is retained in a COS. Every object that can be referenced via a given COS has an object moniker record in that COS. The moniker encapsulates a GUID identity of the object it represents. In the case of an external file object, the moniker provides path information to locate the file and is known as a file moniker.

When a customer **160** wishes to obtain a title from the server **246** (when connected to the preferred network), the system performs a file copy of the title COS **840** to the file system of the user's local workstation **182**. The customer may then proceed to view the downloaded title in the MPS viewer application. This action forces object dereferencing. The local COS manager will be unable to locally satisfy these resulting object request from within the stripped-down title COS **840** at the customer's workstation **182** and will field them in the manner of the Object Broker scenario described previously. Once this title is downloaded to a customer's workstation **182**, the title is thereafter kept up to date with the latest version of the publisher's title objects via on-going connection sessions to the server **246** over time and the MPS Object Broker mechanism operating upon on all point nodes in the distributed system.

a. Object Broker Operation

The MPS COS Object Broker supports two primary services or operations: object request retrieval and object cache updating and title publishing to a server, where the title is stripped bare of objects suitable for end user downloading. In addition, the Object Broker also automates object aging and expunging from its object cache, maintain an object update list on a per title basis, and maintain a list of other object broker servers that have been or can be connected to.

b. Implementation Description

The Object Broker executes as a process upon its host machine, such as the publisher workstation **180**. It operates as a server for both local processes and remote client processes executing on external machines. Access to the Object Broker interface from external machines is made via the Remote Procedure Call (RPC) protocol. In the case of Microsoft Network client-to-server access, the protocol of interaction is MPC. MPC is used for object broker-to-object broker dialog and is encapsulated in the Object Broker; client code is shielded from MPC via the Object Broker public interface. Thus, local machine interprocess access of the Object Broker interface is via the OLE 2 COM convention where proper interprocess marshaling must take place on client/server interactions.

While executing as a process on a server site **242**, the Object Broker is acting as a background task for fulfilling any in-coming object requests, title COS updates, and title COS publish operations. The Object Broker implementation is as a Win32 32-bit executable and as such, takes advantage of Win32 multi-threading so as to service multiple client requests. Access to the Object Broker's hidden file COS object cache also supports multiple process and multiple thread accesses. That is, the COS manager must too support concurrency.

C. Publisher-side Implementation

From a publisher site **102** perspective, the Object Broker offers a unique publish **842** service. Publishing a title COS **840** conveys the title COS to a server site **242**. The Object Broker **546** of the publisher-site machine **180** interacts with the Object Broker **542** of the server-site machine **246**. The COS compound document file, of which the title COS **840** primarily consist of, is transported to the server site **242** where it is stored again as a file in the file system of the server **246**.

The title COS **840** is entered and stored in the superCOS **832** of the server site Object Broker **542**. All objects receive a GUID identity at creation time in the designer environment **194** (FIG. **2**). Objects which have a pre-existing GUID identity update older versions of themselves in the local cache **832** upon collision. Discreet objects are accessible or retrievable in confederation to the server site Object Broker **542** via their GUID identity. The GUID identity is used by the Object Broker **542** as a lookup key into its map of cached objects.

A given published title. COS, such as title COS **842**, also possesses a unique identity (subsumed from the GUID of its root-most object). It remains so uniquely identified at the server site **242** to which it is published **842**. Future attempts to republish the title result in updating the version of it that exist on the server site **242**. The identifying GUID of the title COS is retained at the publishing site **102**.

From the publisher site **102** it is also possible to publish folders of title content as a content folder subCOS **592** (FIG. **11***a*) in a manner similar to that of a title subCOS. Prior to publishing a folder of content, the individual content files must be linked into the respective title. Hence, a complete title may consist of a title COS of objects plus a folder of content objects, where content objects consist of a directory hierarchy of discreet files that have been linked into the title in various usage relationships. A plurality of published content folder subCOSes **856** are shown in the local COS cache **832** of the server site Object Broker **542**.

A published content folder is known by the Object Broker GUID identity of the local machine from which the folder originates (the publisher's machine **180**). It is further possible to use the name of the folder as a lookup key in conjunction to the Object Broker GUID. The server site Object Broker **542** retains archives of the content folder from previous publishings of the folder. It is possible to query the server Object Broker **542** and enumerate all retained archives of a particular published folder.

d. Customer-side Implementation

The primary Object Broker service from the customer site perspective is object retrieval, coupled with local caching of objects, and the periodic expunging of stale objects from the cache, so as to prevent inordinate usage of the end-user's precious hard disk space.

Customers will be able-to download published title COS files from server sites **242**. Attempting to view a downloaded title COS **840** at the customer COS **548** via the MPS Viewer application effects the forcing of object requests. For

example, the COS **840** in use by the Viewer application attempts to load a title object and discovers that it is missing. Consequently, the COS manager then invokes the local Object Broker API to retrieve the object by supplying the absent object's GUID identity to the Object Broker **550**. In this context, local refers to the object broker **550** operating at the site of the customer's workstation **182**. The object's GUID is encapsulated within the object moniker data structure. The root moniker, in turn, is contained in the moniker table (see table **600**, FIG. **12**) within the title COS.

The local Object Broker **550** then attempts to look-up the object, such as object **850**, within its local object cache. Some objects, such as object **848**, are present in the cache (indicated by a bold object circle). On a cache miss, such as for object **850** (shown as a lighter object circle), the local Object Broker dispatches an object request **852** to the Object Broker at a remote server site **242** via RPC or MPC. If discovered remotely, the requested object is propagated to the customers machine and is entered and stored into a local Object Broker cache **834**. The COS manager, on behalf of the Viewer application, then completes the object load from the Object Broker cache **834** and henceforth retrieves the object from there, until a point in time when or if the object becomes expunged from the cache (in which case the remote retrieval sequence is reprised so as to satisfy any new request for the object).

The result of this process is that the title COS **840** can be mastered by a publisher **102**, published to a server **246**, and then periodic updates to the contents of the title (which would perhaps be objects comprising the title itself or a published content folder) could then be made by the publisher **102**. By subscribing to a title, customers could acquire its current contents during log-on sessions while connected to a title server **246**.

A title COS **840** downloaded to a customer's machine **182**, by these mechanisms described, forces retrieval of the most current published content information. This would happen by interactively browsing a title or else by the home delivery of the title, which would be enabled while subscribing to the title.

As briefly mentioned above, home delivery is a special scheduled service where the customer's machine connects to a server site **242** at off-peak hours. The customer's machine automatically acquires the latest published title by retrieving all the objects necessary to view the title and storing the objects in superCOS caches.

e. Server Implementation

The server **246** is the public repository site of published titles and published content folders. A stripped-down published title COS, such as COS **840**, is retained in the file system of the server **246**. Hence the customer workstation **182** can retrieve a title by merely performing a file copy of the title file to their local machine's file system. Published content folders **856** result in the creation of subCOS entries that hierarchically reside under the root Object Broker COS **832**. The objects contained by a content folder, which were discreet files on the publisher site, become COS IStream objects within their respective content folder subCOS. Client end-users cannot directly access the objects comprising content folders.

In either case of title COS objects or content objects, both are ultimately identified by a GUID. The server Object Broker **542** accepts their GUID as a key by which to look them up for object retrieval purposes. The MPS publisher's tool suite provides for assigning GUIDs to content objects by an OLE API at the point in time that they are linked into a title that is under preparation. Additionally, it is possible to key for a content folder.

A title COS can be keyed in confederation to the server Object Broker **542**, by supplying its GUID identity. A title COS **840** is not in any way automatically archived by the server Object Broker **542**. Once created, a title COS **840** and its contained objects tend to be very long lived and are not updated on as frequent a basis as are content objects. Consequently, a publisher site **102** is responsible in totality for title COS archiving. A publisher site **102** must also retain a stub of a published title COS **840**. This stub title COS need only contain the object moniker of the root-most COS object—which encapsulates the GUID of the root-most object. This root-most object GUID identifies the title COS as when it was published to a server Object Broker **542**.

f. COS External Files

Objects of a COS are referenced with a 32-bit locally-scoped object handles. Every object known by a COS has an object moniker entry in that COS. Objects which are actually contained in the COS are placed into IStreams that are hierarchically scoped from the root IStorage of the COS.

In the case of content files as found under a content folder, these are not made to be contained internally in a COS (this is only the case, however, for a publisher site workstation **102**), but instead are linked to a COS by use of a special file link moniker mechanism. Such external files can also be made to have links to other files in a usage relationship manner. All of these links involving external content files cause an object moniker entry to be generated in their respective COS (the subCOS instantiated and contained within the local Object Broker COS) for the link collect ion of items from a specified content folder.

V. VIEWER DETAIL

A. Overview

1. Subsystem and Interfaces

This section describes the viewer component (also known as the Viewer **202**, FIG. **2**) for the MPS architecture. The Viewer **202** is responsible for synthesizing a title into its composed format. This component can be described as providing the run-time view of a title, while the project editor component provides a design-time view of a title.

a. Subsystem and Interface Overview

The viewer component synthesizes the set of objects, which define a title, into a rich multi-media document which is viewed and navigated on the client machine. The synthesis process consists of two basic functions: acquiring the content and composing the content. In the case of a static title (one which does not make use of search objects or links to external content), the content acquisition step is not necessary, as all of the content required to compose the title is already defined in the title. A dynamic title requires the acquisition step to resolve search objects or links, acquire content, and insert it into the title. This process of a acquiring content can be applied successively, appending or replacing previously acquired content in order to update the instance of the title being synthesized. Having acquired the content, the title can now be composed and rendered on the client machine for viewing. The viewer component provides additional interfaces for navigating and managing hyper-text links while a title is being viewed by the client. Several subsystems are used to perform this process of synthesizing a run-time view of a title.

At the top-level there is a general viewer subsystem and interface (IViewer) for initiating the synthesis of a given title. This interface includes methods for instigating the content acquisition process which creates or updates an instance of a title (pressed title or issue). There are also methods for initiating the composition and viewing of one of these instances. Finally, there are methods for storing or archiving one of these instances (back pressing or back issue).

The content acquisition subsystem is invoked by the general viewer subsystem to resolve search objects, retrieve content attracted to the search objects, and insert the retrieved content into the structure of the title instance being created or updated. The acquisition subsystem can be prompted to acquire content for the entire title or an individual section. This capability allows individual sections to be scheduled for update. The content acquisition subsystem traverses the given title or section looking for search objects to resolve. The IResolveMagnet interface is used to retrieve a list of content IDs (path based or GUID based) for each search object. These content IDs are references to the actual content objects (MPML, graphics, audio, video, and so forth) and are inserted into the title instance in the appropriate section, associating that content with that section.

Depending upon the viewer mode, home delivery or browse, the content acquisition subsystem may or may not actually retrieve the content objects referenced by the content IDs. In home delivery mode, the IBBobjectBroker interface is used to retrieve the actual content objects immediately and cache them in the local object store. In browse mode, the actual content is not retrieved until it is required for composition (is to be viewed by the user). When the user is browsing a title interactively, the mechanism for retrieving content may be through the IContentServer or IMediaServer interfaces. These interfaces support the transmission of specific content objects in an efficient manner. For instance, a MPML file may be pre-parsed on the server and only a partially resolved parse-tree transmitted to the client for composing common tags within Infomaps, such as abstracts or table of content entries. An Infomap is a special kind of control that provides automatic display and navigation capabilities. Graphics may be transmitted as progressive renderings, audio as short sound bites, video as first frame, and so forth.

When the viewer subsystem is prompted to compose a given instance of a title, the first step is to create a parse tree which represents the entire structure of the title. The actual parsing of the MPML content can take place on either the server or within the viewer component on the customer workstation. The result is a complete tree for the entire title, including the structure inferred by the title at the top of the tree, and the parse trees for each individual MPML object at the bottom of the tree. The parse tree may or may not be fully resolved with the content depending on the viewer mode. It is at the parse tree creation step that sorting algorithms can be applied to order the content that is placed into the tree. This parse tree exposes an IParseTree interface which is used by the Infomap and OLE dynamic story controls to retrieve the relevant tagged elements from the title structure.

In addition to the parse tree, a form block manager is initialized to manage and track which elements of the parse tree structure are composed onto which pages. This page block manager tracks the page breaks for the content, and provide an IPSF interface for the OLE dynamic story control to identify where in the parse tree to start composing on a given page.

A navigation subsystem provides the INavigate interface of which controls, menus, internal and external scripts, and so forth can be used to manipulate the view of the title. The navigation subsystem uses the page block manager and parse tree to manage the creation, composition, and viewing of pages.

The viewer component also contains a Link Manager subsystem which provides the ILinkManager interface. This interface is used by Infomap and OLE dynamic story controls to register and resolve hyper-text links within the title. These links may be either synthesized (Infomap control), or obtained by a tagged-link provided within some content (dynamic story control).

b. Subsystem and Interface Detail

The subsystems and interfaces provided by the viewer component as well as the interfaces which are required from other components will now be further described.

i) Provided Interfaces

The Viewer provides the following interfaces, all of which are contained in the VIEWDLL DLL:

IViewer—Provides methods for acquiring content, composing, viewing, and archiving titles.

BOOL Open (const CString& filename)

Open the given title file for synthesis by this viewer object. The viewer object assumes that it has the right to modify this file with regards to acquiring content. The creator of the viewer object should assume the responsibility for duplicating the original title file, if necessary.

BOOL ArchiveAs (const CString& filename)

Create an archive of the currently open title into the given filename. This method causes all of the locally cached objects within the client object store which are referenced by the title to be copied into the archive file. This operation produces a completely self-contained title which can be viewed in the distant future without regard to objects having been updated or removed from the local cache or server.

BOOL AcquireContent (BOOL cacheLocal =TRUE)

Acquire content for the currently open title. This method iterates through the entire title resolving active search objects with the server. The content IDs resolved by the search objects are inserted into the proper section, associating the content with the title. By default, the actual objects are downloaded from the server and cached in the local object store (home delivery mode). Providing a FALSE cacheLocal parameter will defer the download of the actual object until it is viewed by the user (browse mode).

BOOL AcquireContent (const CString& sectpath, BOOL cacheLocal =TRUE)

Acquire content for a given section of the currently open title. Identical to the previous method, but reduces the scope of the content acquisition to a given section (full path name). This allows sections to be scheduled for acquisition and update on an individual basis.

BOOL Compose ( )

Compose the currently open title for viewing. This method creates a parse tree for the entire title structure and initialize a view block table for managing the composition of pages. It then identifies the first page of the title, composes it, and displays it to the user. Subsequent navigation through the title causes other pages to be composed and displayed.

CElementNode* GetRootNode ( ) const

Returns the root node of the parse tree for the currently composed title. The IParseTree interface can then be used to locate specific elements within the title. Note that the root node is only available when a title has been composed, otherwise this method returns NULL.

IParseTree—Provides methods for walking the nodes of the parse tree and for extracting a list of specific element nodes from the parse tree.

IPSF—Provides methods for retrieving the next element node and character position for composing the dynamic

story control on a page. This interface is used exclusively by the dynamic story controls for determining the starting point of composition for a given page. The dynamic story control uses the IParseTree interface to retrieve each subsequent node of the tree, until the entire dynamic story control region has been composed or there is no more content to compose. When composition is complete, the dynamic story control marks the ending node and character position with another method of this interface.

CElementNode* GetNextPSFNode (DWORD formID, CElementNode** startNode, DWORD& startPos)

Called by a dynamic story control to retrieve the story node, starting element node, and character position to start composing the page identified by the given formID. The returned node is the story node within the overall title parse tree structure. The startNode is set to the specific element node at which composition should begin. The startPos is set to the character position within that specific element node.

void MarkLastPSFNode (DWORD formID, CElementNode* endNode, DWORD endPos =kNotDoneRendering)

Called by a dynamic story control to mark the element node and position at which composition of the dynamic story flow for the given page has ended. This information is stored in the view block table along with the starting story node, element node, and position. If the end of a story is parsed through and the dynamic story control can continue composing text, it will mark the end of the story with an endpos of kNotDoneRendering and will call GetNextPSFNode to continue composing the next story, if one exists.

INavigate—Provides methods for navigating to various logical areas of the title being composed.

BOOL	GoNextForm ( )
BOOL	GoPrevForm ( )
BOOL	GoNextStory ( )
BOOL	GoPrevStory ( )
BOOL	GoNextSection (BOOL wrapAround =TRUE)
BOOL	GoPrevSection (BOOL wrapAround =TRUE)
BOOL	GoToSection (const CString& sectionPath)
BOOL	GoToForm (const CString& formName)

ILinkManager—Provides methods for registering and resolving links.

BOOL	ResolveLink (DWORD* linkHandle)
DWORD	RegisterLink (const CLink& link, BOOL* isResolved)
DWORD	RegisterLink (const CString& sectionPath)
DWORD	RegisterLink (DWORD contentHandle, BOOL* isResolved)
DWORD	RegisterLink (GUID contentID, BOOL* isResolved)

ii) Used Interfaces

The Viewer uses the following interfaces:

IResolveMagnet—Provides methods for resolving a search object (magnet) to a set of relevant content.

IBBObjectBroker—Provides methods for acquiring an object from the server.

IContentServer—Provides methods for acquiring content objects from the server in an efficient manner (this refers to tagged-text (MPML) objects).

IMediaServer—Provides methods for acquiring multimedia objects from the server in an efficient manner, which include graphics, video, and audio.

ITitle—Provides methods for accessing and manipulating the collections of objects which define a title.

ISection—Provides methods for accessing and manipulating the collections of objects which define a section.

IForm—Provides methods for accessing and manipulating the properties and control properties which define a page.

B. Viewer Structures

1. Basic Title Parse Tree

Referring to FIG. **17**, an exemplary title tree **890** will now be described. A title tree is an in-memory representation of objects of a title in the MP system **100**, wherein the objects are streams and storages in a COS. The title tree is utilized by the viewer component **202** to facilitate the viewing of a title by the customer. The title tree **890** comprises a root node and a series of nodes arranged below the root in a tree format to present a hierarchy of information. A tree is a well known software data structure. Each of the title, the sections, the subsections (if present), and the roots of the stories are the OLE storages, previously described. Each of these storages has a GUID assigned to it. Beneath a story root is a parsed tree representation of content that has been stored to a COS, known as a MPML parse tree. The MPML parse tree represents a parsed version of a structured story in a tree format, wherein any item that is tagged during the authoring phase has a node in the tree (including OLE objects). The MPML parse tree is further disclosed in a copending application also assigned to Microsoft Corporation, entitled "Structured Documents in a Publishing System," previously cited. At the base of the MPML parse tree are nodes, known as leaf nodes, that are the streams that store the data, such as text or embedded objects. Nodes above the leaf nodes are the storage nodes.

The title tree begins with a title root **892**. Associated with the title root **892** is a GUIDa that uniquely identifies the title. Below the root, at the next level of the title tree, are a series of sections. Section 1 is represented by a node **894** and has a GUIDb associated with it that uniquely identifies the section. Section 2 is represented by a node **896** and has a GUIDc associated with it. Section 3 is represented by a node **898** has a GUIDd associated with it.

In this example title tree **890**, Section **894** has a story 1 and a story 2. Story 1 is represented by a root **900** and has a GUIDe associated with it. Story 2 is represented by a root **902** and has a GUIDf associated with it. In this example, the root **900** has a MPML parse tree **908** below it, which is a parsed version of the content identified by GUIDe. Below root **902** is a MPML parse tree **910**.

Section **896** has a story 3 represented by a root **904** having a GUIDg. Below root **904** is a MPML parse tree **912**. Section **898** has a story 4 represented by a node **906** having a GUIDh. Below root **906** is a MPML parse tree **914**.

The title tree **890** will be further referenced in conjunction with the viewer structures described in FIG. **18***a* and **18***b*.

2. View Block Table

Referring now to FIGS. **18***a* and **18***b*, structures used by the Viewer **202** (FIG. 2) in the viewing of a title will be described. When the Viewer **202** is initiated, either a title COS having a root is given to the Viewer, or the Viewer creates a new COS with a moniker. In either case, the Viewer **202** opens the COS and accesses the title. The Viewer **202** then creates a view block table, such as exemplary table **920**, and synthesizes a title tree, such as title tree **890** (FIG. **17**). These data structures are non-persistent.

A view block table is an array of view block lists (described below), wherein there is one list per section of the

title. The exemplary view block table **920** is created by the Viewer **202** by traversing the container portion (the top) of the title tree, beginning at the title root and proceeding down each branch to the subsection level (if present), or the section level if there is no subsection for the section. Each container object is placed in the view block table **920**. Using the title tree **890** as an example, the table **920** has a title entry **922**, corresponding to the root **892** of the title tree, a section 1 entry **924**, corresponding to the node **894**, a section 2 entry **926**, corresponding to the node **896**, and a section 3 entry **928**, corresponding to the node **898**.

The Viewer **202** traverses the view block table starting at the title and proceeds to each section (and subsection, as applicable) as the view blocks are filled in. At each section, the Viewer **202** determines whether the section has a page object. If not, the Viewer cannot display anything, and proceeds to the next section. If there is at least one page in the section, the Viewer determines which page to view. At the same time, in another thread, the Viewer initiates content acquisition. The stories for the current section in the title tree are brought over from the server COS and are instantiated at the Viewer **202**. For a dynamic control within a section, search objects must be resolved to get the stories. The stories are received in the form of a MPML parse tree, and are appended to the title tree below the corresponding section (or subsection, if applicable). A page composition and rendering process can begin as soon as a section having a page and a corresponding story are present at the Viewer **202**. Thus, a story is composed as the Viewer **202** walks the tree.

There are two stories in the first section of the exemplary title tree **890** (FIG. **17**). A view block list is created for each section in the table **920** having a page. The view block list is an object that contains a list of view blocks (one view block per page in the section), and an array of handles to parse trees for the stories in the current section. In the example of FIG. **18**, a view block list **930** is created for the section entry **924**. The list **930** has a story 1 pointer **936** and a story 2 pointer **938** to the respective stories in the title tree **890**. A view block object is created by the Viewer **202** for each page that contains a dynamic story control. The view block basically tracks which story and how far into the story the composition and rendering process has progressed at the end of a page. There can be one or more view block per story.

Each section in the view block table contains page references. The page references are built up in the view block table (from the COS) as page requests are made.

An exemplary view block **940**, with initial field values, is shown in FIG. **18***b*. A beginning story index **956** is initialized to zero since it is the first story in the section. An ending story index **958** is also initially set to zero (start where end). An end story node **960** (to which node the viewer progressed) is initially set to story 1. An end position index **962** (how far through a node) is initially set to zero. A page type **964** is set to the type of page object currently being processed. A new view block is created using the ending information from the previous view block as beginning information. In the example view block list **930**, additional view blocks, view block **942** and view block **944**, were created to complete the rendering of story **936** and story **938**.

As the customer navigates through the title, e.g., by activating a link to another story, a "resolve link" function locates the section for the desired story in the view block table. The resolve link function then calls a "display story" function and passes an index of the story to be displayed. The display story function looks through the view block list of the located section to see if the requested story has already been composed onto a page. If so, the page is displayed to the customer. If not, a compose operation is called to compose the requested story onto a page, followed by showing the composed story to the customer.

3. Title Parse Tree

Referring to FIG. **19** a second exemplary title tree **990**, that is different than the title tree described in conjunction with FIG. **17**, will now be described. This title tree **990** is expanded to include exemplary MPML parse trees and also shows how the tree need not be symmetrical. However, page nodes and control nodes, such as shown in FIG. **17**, are not illustrated in FIG. **19**.

The title tree starts with a title root **992** having a GUIDa. Below the title root **992** are a section A represented by a node **994** having a GUIDb and a section B represented by a node **996** having a GUIDc. Typically, a title is arranged with sections, and some of the sections may have subsections. Stories are inserted into either of the sections or subsections. However, stories may also be placed directly below the title root in the title tree, as exemplified by story C represented by a node **1004** having GUIDg. Section **994** has a subsection represented by a node **998** having a GUIDd.

Below subsection **998** is a story A represented by a root **1000** having a GUIDe. As shown in FIG. **19**, the root **1000** of story A is the root of a MPML parse tree. Below the root **1000** of story A are a head node **1006** and a body node **1008**. The head node **1006** has a leaf node **1018** that, in this example, is the abstract section of the story A at root **1000**. The body node **1008** has a Heading1 (H1) type of style represented by a node **1020**. Below the heading style is a leaf node **1040** having text content for the story. The text content is in the form of a data stream. When instantiated by the Viewer **202** (FIG. **2**), the style above it in the tree, style Heading1, will be applied to the content. Also below body **1008** is a Paragraph1 (P1) style represented by a node **1020**. The Paragraph1 style has a leaf node **1042** below it that is also a data stream of text.

Below the section B node **996** is a story B represented by a root **1002** having a GUIDf. Below the story root **1002** is another MPML parse tree having a head node **1010** and a body node **1012**. The head node **1010** has a table of contents (TOC) leaf node **1024**. The body node **1012** has a Heading2 (H2) style node **1026**, a Wrap Advertising (WA) style node **1028** and a Paragraph2 (P2) style node **1030**. The Heading2 style node **1026** has a leaf node **1044** representing a text content stream. Below the Wrap Advertising style node **1028** is a leaf node **1046** representing an embedded object stream. The Paragraph2 style node **1030** has a leaf node **1048** for a text stream.

As previously mentioned, story C, which is represented by root **1004**, is immediately beneath the title root **992**. Below the story root **1004** is a MPML parse tree having a head node **1014**. Beneath the head node **1014** is an abstract leaf node **1032** containing a stream of an abstract for story C. Also beneath the root **1004** is a body node **1016** having a Heading1 (H1) style node **1034**, a Paragraph1 (P1) style node **1036** and a text stream leaf node **1038**. Further, beneath the Heading1 style node **1034** is a text stream leaf node **1050**. The Paragraph1 style node **1036** further has a text leaf node **1052** below it. As previously mentioned, all leaf nodes are streams. All nodes above the leaf node level of the title tree are storages.

C. Operation

1. Title Viewing Flow

Referring now to FIGS. **20***a* and **20***b*, a title viewing process **1080** will be described. This process **1080** is per-

formed by the Viewer **202** at the customer workstation **182** (FIG. 2) and corresponds with state **334** of the top level process **320** (FIG. **5**).

Beginning at a start state **1082**, the process **1080** proceeds to a state **1084** wherein the Viewer **202** (bbview.exe) is evoked by the customer **160** (FIG. 1) to view a title. In the presently preferred embodiment, the customer can either select a shortcut icon for a title on the Windows 95 desktop that is desired to be viewed, or after selecting the Microsoft Network icon on the Windows 95 desktop, the customer selects a title through the menu system of Windows 95. Of course, titles may also be selected from the CD-ROM **124** (FIG. 1) or other storage **126**. In any case, the bbview.exe program, previously described, is initiated.

Moving to a state **1086**, the process **1080** creates two data structures: a top of a title tree and a View Block Table. The data structures used by the Viewer **202** are non-persistent. This state is the InitTreeAndTable function of the CBViewer class and is part of viewdll.dll. The top of a title tree is the portion from the title root down to the sections (or subsections) included in the title tree, i.e., the containers in the title tree. As an example, for the exemplary title tree **890** (FIG. **17**), the top of the title tree is from the title root **892** down through the section nodes **894**, **896** and **898**. An exemplary View Block Table **920** was described in conjunction with FIG. **18***a*. The View Block Table is initialized to the containers in the top of the title tree, beginning at the title root and proceeding down each branch of the tree to a subsection (if it exists) or a section.

Continuing at a decision state **1088**, the process **1080** determines whether any search objects need to be resolved for the customer selected title. If no search objects are to be resolved, the process **1080** moves to a state **1090** and determines the first section with a valid page to view. Valid pages are those that are either static, or are dynamic and have actual content (stories) to flow into them. That is, a valid page is one that is capable of being rendered and displayed, because the process **1080** knows and understands all of its contents. This task involves walking through the title tree (such as tree **890**, FIG. **17**) to find the first page that fits one of the above criteria. The process **1080** begins with the top node in the title tree **890** and does a depth-first search through all of the sections and subsections. For each section (and subsection), the pages are checked in order to see if each one is valid. As soon as a valid one is found, the section containing that page is returned as "the first section with a valid page to view". This state is the Compose function of the CBViewer class and is part of viewdll.dll.

Proceeding to a state **1092**, the process **1080** navigates to the first valid section. The Viewer **202** keeps track of which section is currently being viewed at any given time. This step sets the initial value for that setting. This is done by taking the section returned by the above state **1090** and writing that value into the "Current section" setting. This state is the GoToSection function of the CBViewer class and is part of viewdll.dll.

Continuing at a state **1094**, the process **1080** determines which page to view. The process **1080** also keeps track of which page is currently being displayed at any given time. This state **1094** sets the initial value for that setting. Within the current section, the process **1080** finds the first valid page, i.e., the first page that is actually capable of being rendered and displayed. This procedure is performed using the same algorithm used in state **1090** above—the process **1080** walks through the pages in sequence, looking for either a static page, or a dynamic page that has content to be placed. This is a somewhat redundant with the code in the state **1090** above, but that is because this is separate code used whenever the customer interactively navigates to a section. This state is the ResolveForm function of the CBviewBlockList class and is part of viewdll.dll.

The process **1080** proceeds through an off-page connector A **1096** to state **1120** on FIG. **20***b*. Returning to the decision state **1088**, if the title contains search objects that must be resolved, the process moves to a state **1098** wherein a search object resolution thread is spawned. A thread is single path of execution within a process, and a process can initiate multiple threads. Using the preferred Windows 95 multi-threading capability this thread resolves the search objects concurrently with the states **1090–1094**. This state is part of the viewdll.dll.

In the thread started at state **1098**, the process **1080** proceeds to a decision state **1100** to determine if there is a section (or subsection) in the title tree to process. If not, the thread ends at an end state **1110**. If there is a section, as determined at state **1100**, the process **1080** moves to a decision state **1102** and determines if there are any search objects in the current section. If so, the process **1080** proceeds to state **1104** to resolve all the search objects in the current section. This state is part of the ircl.dll. Progressing to state **1106**, the process **1080** adds all the content hits (the content objects found by the search objects) to the container of the current section. This state is part of the viewdll.dll. At the completion of state **1106**, or if decision state **1102** determined that no search objects were in the current section, the process **1080** advances to a state **1108** and attempts to access a next section in the title tree. The process **1080** then loops back to the decision state **1100** to determine if there is a next section in the tree. If all the sections have been processed, the thread completes at the end state **1110**. This state is part of the viewdll.dll.

Continuing now at state **1120** on FIG. **20***b*, the process **1080** composes and displays the page (determined at state **1094**, FIG. **20***a*) to view. This state is the DisplayView function of the CBViewBlockList class and is part of viewdll.dll. State **1120** invokes state **1122** to spawn a content acquisition thread to acquire content objects needed by the state **1120** to compose and display the current page. State **1122** is part of the viewdll.dll.

Moving to a decision state **1124** in the content acquisition thread, the process **1080** determines if there is a section (or subsection) in the title tree to process. If not, the thread ends at an end state **1134**. If there is a section, as determined at state **1124**, the process **1080** moves to a decision state **1126** and determines if there is content in the current section, including the content retrieved by the search objects at state **1106** above. If so, the process **1080** proceeds to state **1128** to acquire the parse tree for each content object in the section. This state is part of the viewdll.dll. Moving to state **1130**, the process **1080** appends the acquired content parse trees to the title tree. This state is part of the viewdll.dll. At the completion of state **1130**, or if decision state **1126** determined that there is no content in the current section, the process **1080** advances to a state **1132** and attempts to access a next section in the title tree. The process **1080** then loops back to the decision state **1124** to determine if there is a next section in the tree. If all the sections have been processed, the thread completes at the end state **1134**. This state is part of the viewdll.dll.

To facilitate incremental delivery of a content object, the MP system **100** uses a remote proxy object (not shown). The proxy object allows the system to break a single object abstraction into multiple objects within the distributed object system. For instance, in the case of text, a title or section

references a text proxy object, which in effect represents a single story. The proxy object is used to separate the single story into multiple objects to allow the incremental delivery of the story. If the story was treated only as a single object, then a very large story would require complete transmission before displaying a single part of it. By use of a proxy, the story can be separated into multiple pieces and can then be retrieved incrementally. The title or section still interacts with the story object as a single entity. Proxy objects can also be utilized with other types of content, such as images and audio.

Returning to the state **1120**, the Viewer **202** utilizes the one or more content parse trees that are appended to the title tree from state **1130** and the view block table and view blocks (described in conjunction with FIGS. **18***a* and **18***b*) to compose and render the page. For each control on the page, process **1080** calls the compose method (previously described).

At the completion of state **1120**, the process **1080** proceeds to a state **1136** and waits for a user action, such as clicking on a caption button control or a picture button control on the displayed page. Proceeding to a state **1138**, the process **1080** interprets and processes the user action. This state is part of the viewdll.dll. Moving to state **1140**, the process **1080** causes the Viewer **202** to navigate to a part of the title indicated by the user action, such as a different section. This state is part of the viewdll.dll. Advancing to state **1142**, the process **1080** determines which page to view. State **1142** is identical to state **1094** previously described. This state is the ResolveForm function of the CBViewBlockList class and is part of viewdll.dll. Moving to state **1144**, the process **1080** composes and displays the page to view in an identical manner to that of state **1120**. This state is the DisplayView function of the CBViewBlockList class and is part of viewdll.dll. At the completion of state **1144**, the process **1080** loops back to state **1136** to wait for the next user action. Process **1080** continues until a user action of "exit title" is selected.

2. Object Retrieval Flow

Referring now to FIG. **21**, an object retrieval process **1170** will be described. This process **1170** is performed by the Viewer **202** at the customer workstation **182** (FIG. **2**) and corresponds with state **334** of the top level process **320** (FIG. **5**).

When a title is viewed, the Viewer opens a title file (having a file extension of .ttl) which represents the title. This title file is a COS file. Typically in the online scenario, this would be a skeleton title (i.e., a COS file which contains only a root moniker and no actual objects). A local superCOS is a COS file which contains subCOSes and is used to cache objects on the customer workstation **182** which have been remotely retrieved from the MSN data center **242**. As long as these cached objects are not out of date or flushed, the local Object Broker is able to quickly provide that object the next time it is requested rather than retrieving it from the MSN data center **242** again.

The object retrieval process **1170** is utilized anytime the Viewer **202** attempts to instantiate an object which is persistently stored in the COS as a single object. The Viewer **202** first looks in the local COS being viewed. If the object is not present, it asks the Object Broker to look in the local SuperCOS (at the customer workstation **182**) of cached objects. If the object is not there, the Object Broker preferably attempts to acquire the object from the server **246** at the data center **242** (FIG. **3**).

Beginning at a state **1172**, the object retrieval process **1170** works in conjunction with the title viewing process **1080** (FIG. **20***a* and **20***b*) when the Viewer **202** (bbview.exe) is evoked by the customer **160** (FIG. **1**) to view a title. Moving to a state **1174**, process **1170** attempts to access an object given an object handle from the Viewer **202**. This state is the GetObject function of the IObjectStore class and is part of cos.dll. Proceeding to a decision state **1176**, process **1170** checks to see if the object is in a local COS **1182** for the current title. If so, process **1170** advances to a state **1178** to instantiate object **1184** from an object stream **1180** in the local COS **1182**. This state is part of the cos.dll. The object instance **1184** is then utilized by the title viewing process **1080**.

Returning to the decision state **1176**, if the object is not in the local COS **1182**, process **1170** moves to a state **1190** and requests the object from the customer workstation Object Broker given an object GUID. This state is part of the objbrk.dll. Continuing at a decision state **1192**, process **1170** checks to see if the object is in a local superCOS **1196** on the customer workstation **182**. If **80**, process **1170** moves to a state **1194** to instantiate the object **1184** from the object stream **1180** in the superCOS **1196**. This state is part of the cos.dll. The object instance **1184** is then utilized by the title viewing process **1080**.

Returning to the decision state **1192**, if the object is not in the local superCOS **1196**, process **1170** moves to a state **1200** and preferably makes a remote request to the data center **242** (FIG. **3**) for the object. This state is part of the objbrk.dll. Continuing at a decision state **1202**, process **1170** checks to see if the object requested from the data center **242** is retrieved. If so, process **1170** advances to a state **1204** to cache retrieved remote object **1206** in the superCOS **1196** as an object stream **1180**. This state is part of the cos.dll. The retrieved remote object **1206** is then utilized by the title viewing process **1080**.

Returning to the decision state **1202**, if the requested object is not retrieved from the data center **242**, process **1170** moves to a state **1212** to report an "object not found" exception to the Viewer **202** and terminate the object retrieval process **1170**. This state is part of the objbrk.dll.

VI. CONCLUSION

This section summarizes benefits provided by the present invention. In the MP system, a content provider has a lot of flexibility to choose how a customer will view a story. In addition, the MP system is device independent in that the tagged content can be displayed with high quality on many different devices. For example, a content provider can create a title just once, but the title can be viewed on a VGA screen with one column, a printer with many columns, a small screen personal digital assistant (PDA), an interactive television (ITV) system, a fax machine, or a notebook computer. Different styles can be applied to each of these devices so that the displayed content is formatted appropriately.

Moreover, separating the content and design in the MP system enables sending or distributing stylized high-quality publications over low-speed communications links. This results from the fact that the design and style sheets of many titles remains fairly static while only the content changes regularly. The MP system does not need to send large design descriptions and style sheets to customers' computers unless the designs or styles change. Content can typically be transmitted quickly since it consists of tagged components, not the actual pages and controls themselves. Thus the separation of design and content eliminates much of the communication overhead in an electronic publishing environment.

Further, the MP system supports standards such as Microsoft, Word and Standard Generalized Markup Lan-

guage (SGML) to ensure that the content provider's investment in existing tools can be fully leveraged. The MP system also reads standard HyperText Markup Language (HTML) documents so that existing HTML documents can be easily converted to more sophisticated applications. Additionally, through support of the OLE standard, tools that supports OLE server capabilities can be used to create content embedded in an MPS title. By supporting additional standard file formats, the MPS can also accommodate other tools (for example high-end graphic applications).

In addition to the advantages listed above, the MP system also has other advantages that differentiate this system from other on-line publishing systems. For example, graphic designers can work on the title and page layouts, while authors create content objects. There is a clean separation of responsibilities, with separate tools used by each professional.

Also, new content does not need to be laid out by a designer before it can be published. It can be uploaded to the distribution point and downloaded to customers' machines as soon as the object is completed, since the rendering is automatically done on the consumers' machines based upon the designs in the title's page layouts. Also, since no rendering has been done prior to downloading the title and objects to the consumer's machine, the appearance of a new piece of content does not force the system to re-download any other items.

As stated above, the styles contained in every style sheet are predefined by the MP system authoring program. This program, termed the MPS Document Editor, has the special capability of producing documents formatted in Multimedia Publishing Markup Language (MPML). The MPML is a form of an SGML, but has formatting commands unique to the MP system. Markup languages which are well known in on-line networks identify portions of documents by embedded tags. In an MPML document, there is one MPML tag per document portion and each tag is mapped to a style that is found in a style sheet.

Although the invention has been described with reference to specific embodiments, the description is intended to be illustrative of the invention and is not intended to be limiting. Various modifications and applications may occur to those skilled in the art without departing from the true spirit and scope of the invention as defined in the appended claims.

We claim:

1. An electronic publication system, comprising:

a network server having a publication storage area;

at least one publisher computer coupled to the network server having a designer comprising a project editor configured to manage tiles, containers and objects; a page editor configured to generate title layout pages; a style sheet editor configured to edit style sheets; an object editor capable of creating search objects; and a word processor capable of creating content comprising tagged, hypertext documents;

the designer on the at least one publisher computer configured to create a title comprising at least one of the title layout pages and where the title layout pages are linked and separated from the content; and

a viewer on a viewer computer coupled to the network server configured to retrieve the title from the network server's publication storage area, where the viewer downloads the title layout pages and the content as a displayable title, and the title layout pages and the content are stored in a memory area on the viewer computer.

2. The system of claim 1, where the content comprises compound documents.

3. The system of claim 1, where the content includes text.

4. The system of claim 1, where the title layout pages comprises a plurality of page and control objects.

5. The system of claim 1, where the title comprises an application.

6. The system of claim 1, where the title comprises a service.

7. The system of claim 1, where the viewer retrieves a portion of the title layout pages and a portion of the content for rendering.

8. The system of claim 1, where the content are modified independently of the title layout pages.

9. The system of claim 1, where the title layout pages are modified independently of the content.

10. The system of claim 1, where the content is progressively rendered.

11. The system of claim 1, where the generation of the title layout pages are performed on a first workstation and the rendering of the title layout pages are performed on a second workstation.

12. The system of claim 1, where the viewer includes a query component for retrieving content matching a selected query criteria.

13. The system of claim 1, where the publisher computer includes a structured storage for storing the title.

14. The system of claim 1, where the viewer computer includes a structured storage for storing the received title.

15. The system of claim 14, where the title layout pages in the viewer structured storage is replaced when new title layout pages for the are received from the publication storage area on the network server.

16. The system of claim 1, where the title layout pages comprises a plurality of layout objects.

17. A multimedia publication system, comprising:

a publisher computer coupled to the network server comprising a designer environment having a project editor configured to manage tiles, containers and objects; a page editor configured to generate title layout pages; a style sheet editor configured to edit style sheets; an object editor configured to create search objects; and a content editor configured to edit content comprising tagged, hypertext documents;

a project created by the publisher computer's project editor;

a plurality of titles and content folders created by the publisher computer;.

layout objects created by the page editor the style sheet editor and the project editor;

content objects created by the content editor;

a server connected to the publisher computer having a storage area for storing the project received from the publisher computer;

a customer computer connected to the server having a viewer for displaying the project received from the server storage area, where the content objects and the layout objects of the title are together rendered as displayable portions of the project; and

an automatic tracking system configured to track changes in the content objects and the layout objects.

18. The system of claim 17, where the designer environment additionally includes an identification process for assigning a unique identifier to each one of the content objects.

19. The system of claim 17, where the viewer additionally includes an object retrieving component for retrieving the stored objects.

20. The system of claim 19, wherein the layout objects and the content objects are retrieved only once from the server storage area regardless of the number of times either the layout or the content is repeated in the title.

21. The system of claim 17, where the server storage area stores a plurality of titles.

22. The system of claim 17, where at least one of the content objects is a story.

23. The system of claim 17, where at least one of the content objects is a picture.

24. The system of claim 17, where only a portion of the content objects are received and displayed by the viewer.

25. The system of claim 17, where the viewer includes a query process for retrieving the content objects matching a selected query criteria.

26. The system of claim 17, where the identification of the content objects is unique in the system.

27. The system of claim 17, where the content objects are shared between the titles.

28. The system of claim 17, where each content object has a unique identifier.

29. The system of claim 17, where the content objects are progressively rendered.

* * * * *

APPENDIX G

U.S. Patent No. 6,134,584 to Chang *et al.*

US006134584A

United States Patent [19]

Chang et al.

[11] Patent Number: 6,134,584

[45] Date of Patent: Oct. 17, 2000

[54] **METHOD FOR ACCESSING AND RETRIEVING INFORMATION FROM A SOURCE MAINTAINED BY A NETWORK SERVER**

[75] Inventors: **Sih-Pin Chang**, Old Tappan, N.J.; **Ephraim Feig**, Chappaqua, N.Y.; **Thomas Yu-Klu Kwok**, Washington Township, N.J.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **08/975,865**

[22] Filed: **Nov. 21, 1997**

[51] **Int. Cl.**[7] **G06F 15/16**; G06F 15/173

[52] **U.S. Cl.** **709/219**; 709/225; 709/229

[58] **Field of Search** 709/219, 225, 709/229

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,727,164	3/1998	Kaye et al.	705/28
5,754,655	5/1998	Hughes et al.	380/24
5,768,528	6/1998	Stumm	709/231
5,796,952	8/1998	Davis et al.	709/224
5,878,228	3/1999	Miller et al.	709/235
5,889,863	3/1999	Weber	380/25

OTHER PUBLICATIONS

Rosenfeld et al; "Automated Filtering of Internet Posting" Online, vol. 18, No. 3, pp. 27–30.

Jaeger et al. "Preserving Integrity In Remote File Location and Retrieval"; IEEE 1996, pp. 53–63.

Haskin, David "Traveling Software's Web Ex 20 Brings The Internet to you", Computer Shopper Sep. 1997, v. 16, No. 9, p. 527(1).

Hastings, Bryan "Web Ex: Download now Hyperbrowse Later", PC World, Oct. 1996, vol. 14, No. 10 p. 98 (1).

Chernicoff, David "Web Ex Sifts Through Online Information Glut", Window Sources Apr. 1997, vol. 5, No. 4, p. 92 (1).

Williams, Dennis "Web Ex Browse Internet Offline" Lan Times, Sep. 30, 1996, vol. 13, No. 22, p. 18(1).

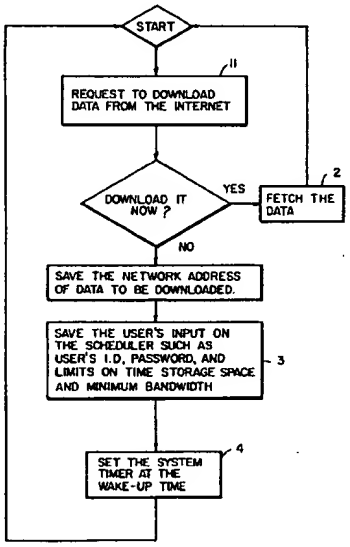
Primary Examiner—Krisna Lim

Attorney, Agent, or Firm—Thomas A. Beck

[57] **ABSTRACT**

A method and system is disclosed for scheduling data download, such as web pages, databases or softwares, over a network such as the internet. The method includes the steps of (a) initiating the data download request and the user input interfaces; saving the requesting computer system's network address; (b) fetching and saving those web pages, databases or softwares' source entities and their corresponding network addresses for the upcoming data download; (c) fetching and saving from the user's input on the schedules, user ids and passwords, disk directories, limits on bandwidth, downloading time and allocated storage space for the data download; (d) setting the system timer at the wake-up time according to the data download schedules; (e) automatically turning on the requesting computer system according to the system timer and dialing up to connect to the network; (f) accessing the download data's network address and transmitting the data to the requesting computer system; (g) receiving and storing the requested web pages, databases or softwares in the requesting computer system; (h) interrupting a particular data downloading if the downloading time exceeding the user's earlier input downloading time limit or if the allocated download storage space reached the user specified limit; (i) automatically disconnecting from the network; (j) automatically rescheduling the data download in some other time if the previous data download is not successful or the network is too busy; (k) automatically turning off the requesting computer system. In this manner the requester doesn't need to keep the requesting computer system power on till the upcoming download actvities.

22 Claims, 3 Drawing Sheets



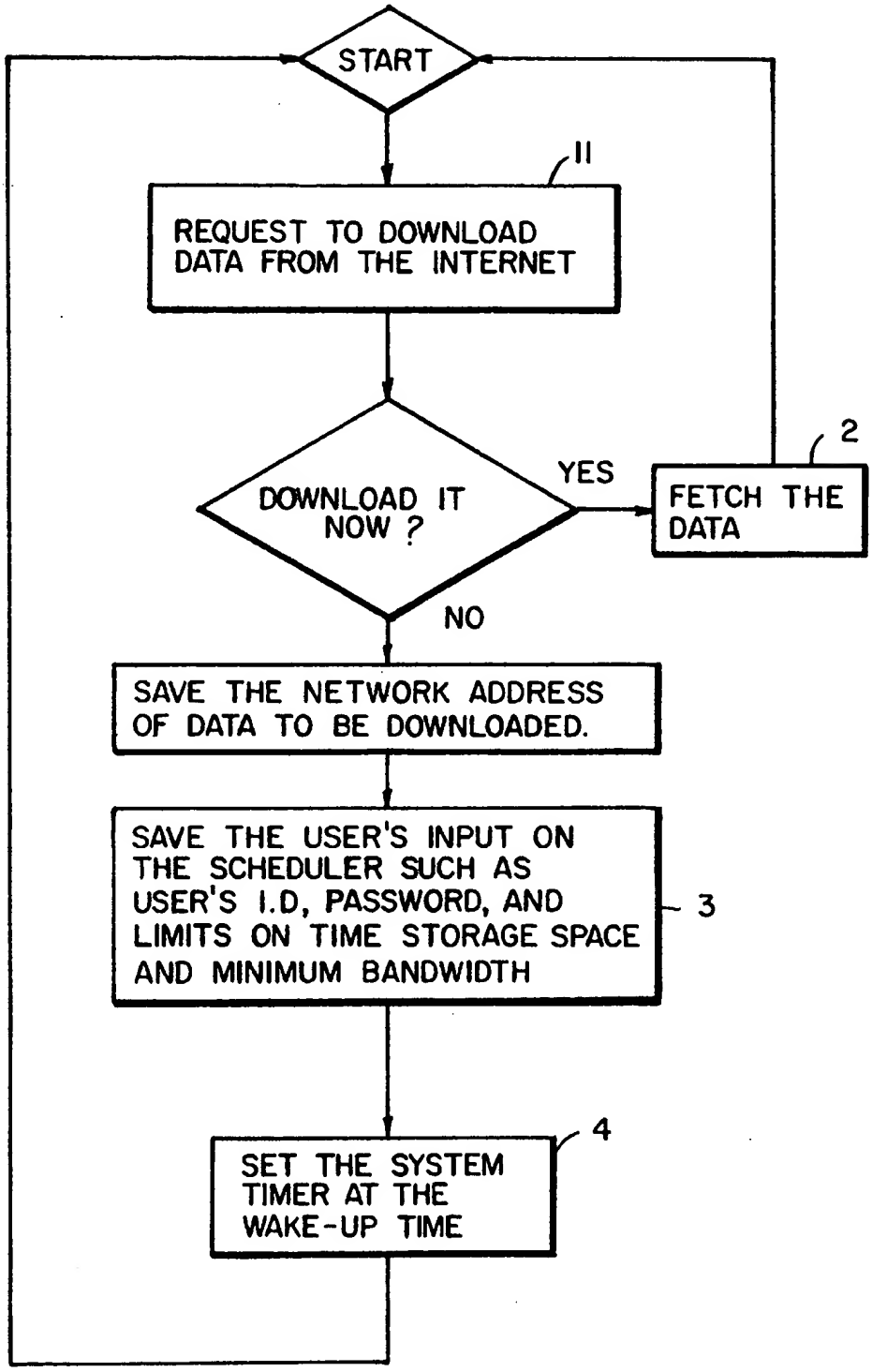

START
11
REQUEST TO DOWNLOAD DATA FROM THE INTERNET
DOWNLOAD IT NOW ?
YES
2
FETCH THE DATA
NO
SAVE THE NETWORK ADDRESS OF DATA TO BE DOWNLOADED.
SAVE THE USER'S INPUT ON THE SCHEDULER SUCH AS USER'S I.D, PASSWORD, AND LIMITS ON TIME STORAGE SPACE AND MINIMUM BANDWIDTH
3
4
SET THE SYSTEM TIMER AT THE WAKE-UP TIME

FIG.1

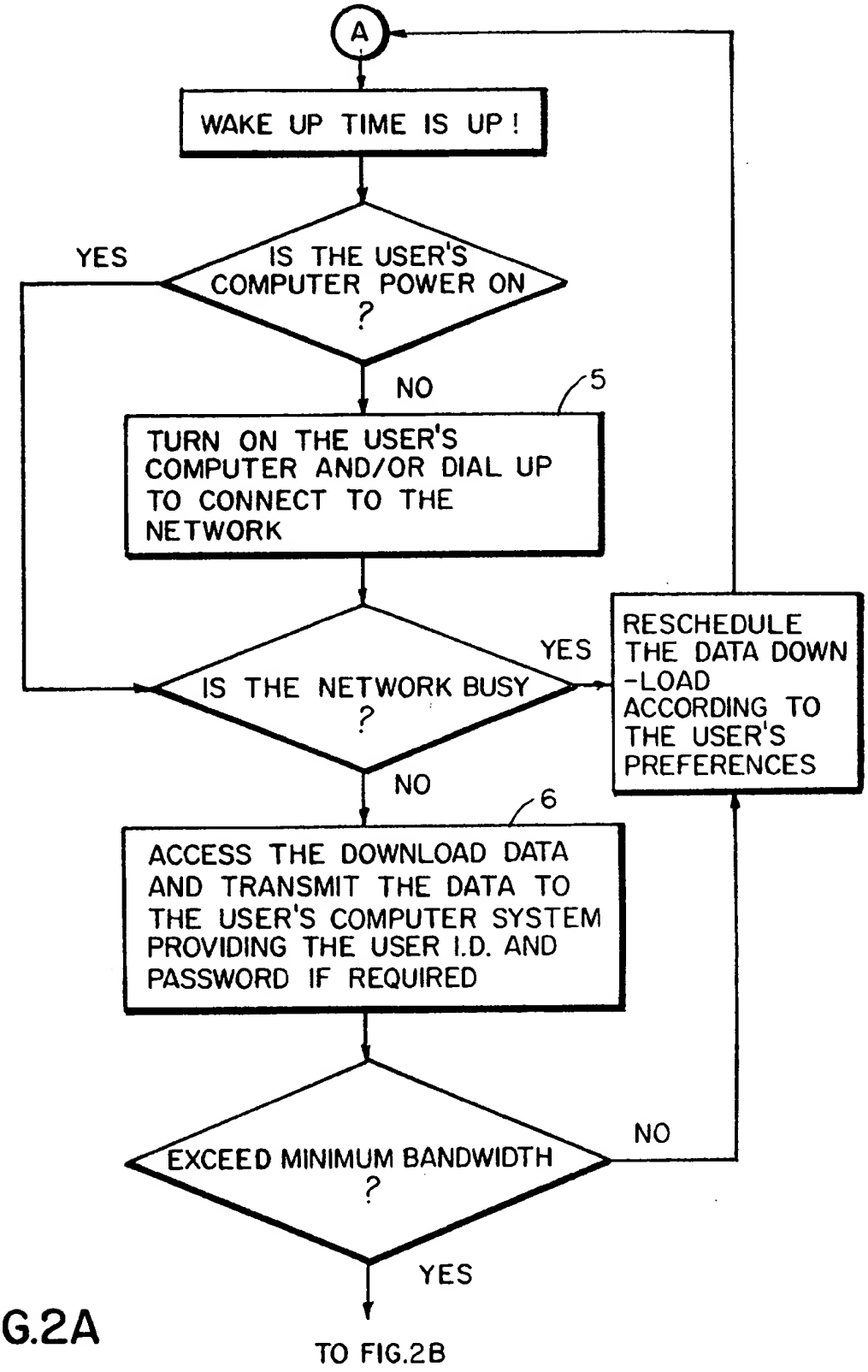

A
WAKE UP TIME IS UP!
IS THE USER'S COMPUTER POWER ON ?
YES
NO
5
TURN ON THE USER'S COMPUTER AND/OR DIAL UP TO CONNECT TO THE NETWORK
IS THE NETWORK BUSY ?
YES
RESCHEDULE THE DATA DOWN -LOAD ACCORDING TO THE USER'S PREFERENCES
NO
6
ACCESS THE DOWNLOAD DATA AND TRANSMIT THE DATA TO THE USER'S COMPUTER SYSTEM PROVIDING THE USER I.D. AND PASSWORD IF REQUIRED
EXCEED MINIMUM BANDWIDTH ?
NO
YES
TO FIG.2B

FIG.2A

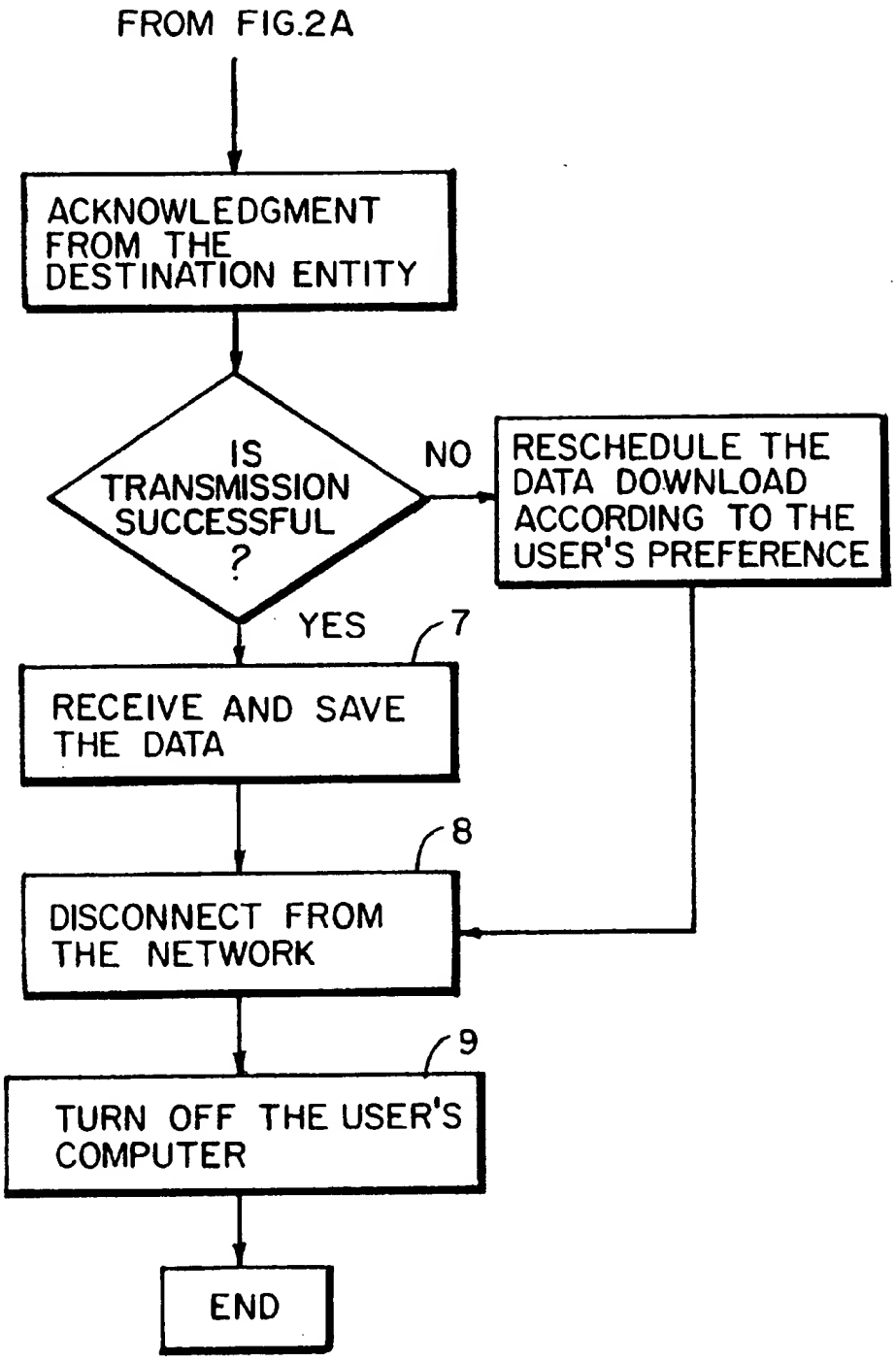

FROM FIG.2A
ACKNOWLEDGMENT FROM THE DESTINATION ENTITY
IS TRANSMISSION SUCCESSFUL ?
NO
RESCHEDULE THE DATA DOWNLOAD ACCORDING TO THE USER'S PREFERENCE
YES
7
RECEIVE AND SAVE THE DATA
8
DISCONNECT FROM THE NETWORK
9
TURN OFF THE USER'S COMPUTER
END

FIG.2B

METHOD FOR ACCESSING AND RETRIEVING INFORMATION FROM A SOURCE MAINTAINED BY A NETWORK SERVER

FIELD OF THE INVENTION

This invention relates to data communications networks and, in particular, to methods and apparatus for accessing and retrieving information from a database, documents, files or web pages maintained by a network server. The methods and apparatus of this invention are particularly useful for scheduling the data download from the World Wide Web (WWW) without keeping the requesting computer system power on all the time till the upcoming download activites. The methods and apparatus of this invention also allow the user to download data from web sites requiring user id and password by supplying the previous stored user id and password on the requesting computer system.

BACKGROUND OF THE INVENTION

An internet user typically employs a browser to access the WWW. The most popular browsers are Netscape's Navigator and Microsoft's Internet Explorer. Often a user wishes to download data from various sites. The user may be linked to these sites during certain periods of the day in which either the internet of the server at the linked sites may be very busy. Downloading at such times could take a very long time. The user may not wish to tie up his browser, or the computer in which the browser is housed, for such a long time. In addition, the connections may require connect charges billed on a per minute basis.

The widespread availability of WWW phones, Personal Data Assistants (PDAs), and Windows-based CE machines with internet connectivity are expected to soon provide internet access capability to larger portions of the earth's population, thereby making efficient techniques to access WWW pages (web pages) even more desirable.

For many users an internet connection made at home some time in the middle of the night may be lower in cost than a connection made during peak phone-rate hours while travelling.

At present there exist several techniques that are known to the inventor for indicating specific web pages to be downloaded at a later time. These techniques download the requested pages to the same (requesting) machine at a later time, for example at night when phone rates are lower and internet traffic is reduced.

There also exist so-called push technology schemes, such as one known as Pointcast™, that periodically download information from certain sites to a given data processor. A user can schedule, for example, news, stock, and/or weather information to be downloaded at specific times or at specific intervals. However, these techniques also download the requested information to the requesting data processor.

Other techniques, such as one known as Webwhacker™, enable a user to make a local copy of a web site, and allow the user to specify a number of links (i.e., Hyperlinks) to follow and download.

A technology available from the assignee of this patent application, referred to as ARTour WebExpress™, allows a user to browse the web more asynchronously than is possible with current browsers. For example, using conventional WWW browsers such as Netscape Navigator™ 3.0 or Internet Explorer™ 3.0 the user can scroll a current page while a next page is being downloaded, thereby providing a

degree of asynchronous access. The WebExpress™ technique takes this one level further by allowing the user to continue to specify links (Hyperlinks) to fetch while previously specified pages are being fetched. These requests are queued by in a local buffer and the pages are fetched in a sequential manner. When the requested pages are available on the local machine, the user is made aware of it by a suitable signaling mechanism.

A proxy server is a World Wide Web server that acts as the sole web server for an entire domain, or for those client computers that are placed behind a firewall (i.e., a logical block between the clients and the rest of the internet). The proxy server typically resides at the firewall and intercepts all web requests originating from clients within the firewall. If a given web page request is not in the proxy server's access control list, the request is processed normally and the retrieved web page is sent back to the requesting client. If, however, the requested web page or web site is on the control list, the client instead receives a message indicating that the URL is not accessible or is not valid.

A proxy server can improve a network's performance by functioning as a caching server. Using its cached web pages, the proxy server will serve already-accessed web pages to requesting clients without requiring outside access to the internet. For example, consider a case of an environment where n client computers access the same web page, wherein each client computer outputs the address (URL) of the web page to be accessed. Without the use of the proxy server, n separate requests for the web page are initiated, and n separate copies of that same web page are retrieved and returned to the client computers.

Using a proxy server, however, the same n web page requests are handled more efficiently. Only the first request to reach the proxy server actually causes that web page to be retrieved from the WWW server, and only if that web page is not already stored in the proxy server's cache. When retrieved, the web page is sent back to the requesting client computer, and is also cached on the proxy server's hard disk. The remaining n–1 clients that request that same web page are then served instead from the proxy server's cache, thus avoiding unnecessary duplicated requests and delays.

An article by L. B. Rosenfeld and M. P. Holland (Online, Vol. 18, no. 3, pp 27–30, May 1994) discusses an internet filtering system which automatically downloads postings daily.

An article by T. Jaeger and A. D. Rubin (Proc. of Symp. on Network and Distributed System Security, IEEE Computer Soc. Press, pp 53–63, 1996) discusses an automated file location and retrieval system which retrieves files and software across the internet.

A web page from Traveling Software (http://www.travsoft.com/press_room/webex.htm) discloses a software downloading package known as WebEx which allows a user to select which web sites to download as well as when to begin the downloading operation. For this method to work, the receiving client must be powered on. Also the downloading will occur no matter what bandwidth is established between the server and the receiving client. This may not be desirable, as in the case where costs are proportional to the time spent for the link to be operational, and when the link is of very low bandwidth, thereby making the download expensive.

However, none of the existing techniques that are known to the inventors enable web pages and other data to be downloaded to this or anoter machine at some other specified time, whether on not that machine is powered on or not.

Nor do any of the existing techniques that are known to the inventors enable data to be downloaded only if a minimum prescribed bandwidth between the server and the receiving client is obtained.

OBJECTS AND ADVANTAGES OF THE INVENTION (TK)

It is an object and advantage of this invention to provide an improved method and apparatus for downloading information from a server that overcomes the foregoing and other problems.

It is an object and advantage of this invention to provide a method and system for scheduling the data download from the World Wide Web (WWW) without keeping the requesting computer system power on all the time till the upcoming download activites. The methods and apparatus of this invention also allow the user to download data from web sites requiring user id and password by supplying the previous stored user id and password on the requesting computer system.

SUMMARY OF THE INVENTION

The foregoing and other problems are overcome and the objects of the invention are realized by methods and apparatus in accordance with embodiments of this invention.

A method and system is disclosed for scheduling data download, such as web pages, databases or softwares, over a network such as the internet without keeping the computer system power on all the time till the upcoming data download activities. The method includes the steps of (a) initiating the data download request and the user input interfaces; saving the requesting computer system's network address, the requesting computer system being either connected to or disconnected from the network; (b) fetching from the internet (the network connected requesting computer system) or from the user's input (the network disconnected requesting computer system) and saving those web pages, databases or softwares' source entities and their corresponding network addresses (e.g. the web site URL on the distant server) for the upcoming data download (which web pages and their linked sub level pages are to be downloaded); (c) fetching and saving from the user's input on the data download schedules (when the data download is to take place) and user id and password if required for those web pages, databases or softwares for the upcoming download; and other user's options and choices, such as where the downloaded data is to be stored (i.e. cache or disk directory), limits on downloading time and allocated storage space, and automatically rescheduling another data download for previous unsuccessful data download; data downloading either scheduled all at the same time in sequent or individually at different time; (d) setting the system timer at the wake-up time according to the data download schedules; (e) automatically turning on the requesting computer system according to the system timer if the requesting computer system is not turned on, and dialing up to connect to the network; generating a network busy signal to the Internal Data Download Schedule for rescheduling the data download according to the user's input of options and choices if the network is busy; (f) accessing the download data's network address and transmitting the data to the requesting computer system; providing the user id and password if required; (g) receiving and storing the requested web pages, databases or softwares in the requesting computer system for subsequent use by the user; transmitting successful and unsuccessful download messages from the destination entity to the requesting computer system; accordingly generating successful or unsuccessful data download messages for the user and unsuccessful data download signals to the Internet Data Download Scheduler to schedule another try of data download according to the user's input of options and choices; the user can access all these messages when the user logs on the requesting computer system; (h) interrupting a particular data downloading (transmitting and receiving) if the downloading time (estimated or actual) exceeding the user's earlier input downloading time limit or if the allocated download storage space reached the user specified limit; (i) automatically disconnecting from the network; (j) automatically rescheduling the data download in some other time, such as every hour afterwards, according to the user's earlier input choice if the previous data download is not successful or the network is too busy; (k) automatically turning off the requesting computer system. In this manner the requester doesn't need to keep the requesting computer system power on for the upcoming download actvities.

The most important advantage of this invention is that the requesting computer system does not have to be power on all the time till the upcoming download activities.

Other advantages of this invention include the abilities of allowing the user to schedule data download from those web sites requiring an user id and password, and to specifying the upper limits on the download time and the allocated storage space for downloaded data.

In one embodiment the step of initiating includes steps of generating a web page download command and transmitting the web page download command to the destination entity. This may be accomplished over the network, or over another network, such as an intranet, that connects the initiating and destination entities. In this case the step of fulfilling includes initial steps of formulating, in response to receiving the web page download command at the destination entity, a network web page request message and transmitting the network web page request message from the destination entity to the web page source entity. A confirmation message may be sent to the initiating entity to confirm the receipt of the web page download command.

In another embodiment of this invention the step of initiating includes steps of generating the network web page request message that includes the user id and password if required and transmitting the network web page request message from the initiating entity to the web page source entity. In this case the web page source entity checks the user id and password if required before transmits the requested web page(s) to the destination entity at the requesting computer system network address.

The step of initiating includes a preliminary step of responding to a signal from a user through a user interface, such as by redefining mouse clicks when interacting with a web browser, such that the signal indicates that a specified web page is to be downloaded to and stored in the destination entity, as opposed to being fetched and displayed in a conventional manner. In this embodiment the step of responding includes a step of prompting the user to enter information for specifying at least one parameter related to downloading the web page, and/or includes a step of retrieving at least one user default parameter related to downloading the web page.

In a preferred embodiment the web page download command sent by the initiating entity includes a plurality of fields, including fields intended to specify: at least one user download preference; and at least one postprocessing operation to be performed on a received web page. The at least

one user download preference includes at least one of: a number of web page levels to download; whether to download graphical data; a number of permissible retries to download a web page; and an interval between the retries. The at least one postprocessing operation includes at least one of: whether to decompress a received web page; whether to virus scan a received web page; and whether to print a received web page.

The step of receiving and storing the requested web page in the destination entity includes a step of writing data into an index entry associated with the received web page. The index entry is comprised of a plurality of fields, including fields intended to specify: the first and second network addresses, and a link summary of the web page. The index entry fields further specify at least one of: a time that the web page was downloaded; a number of bytes that were downloaded; a time that the web page download command was received by the destination entity; a number of retries that were required, if any, to download the web page; and an error report.

In a preferred embodiment of this invention the method includes a capability to transmit a cancellation message from the initiating machine to the destination machine. In response to receiving a cancellation message the destination machine one of terminates an on-going web page download, or deletes an already downloaded and stored web page, as well as the index information associated with the stored web page.

In a preferred, but not limiting embodiment, the network includes the internet, and the web page source entity is a WWW server compliant with conventional and/or extended HTTP protocols.

BRIEF DESCRIPTION OF THE DRAWINGS

The above set forth and other features of the invention are made more apparent in the ensuing Detailed Description of the Invention when read in conjunction with the attached Drawings, wherein:

FIG. 1 is a logic flow diagram showing how to set up the Internet Data Download Scheduler;

FIG. 2 is a logic flow diagram showing a method executed by the Internet Data Download Scgeduler when the system timer is up at the scheduled wake-up time. FIG. 2 is a logic flow diagram showing a method executed by the IDP of FIG. 1; and

DETAILED DESCRIPTION OF THE INVENTION

The following description of this invention is made in the context of methods and apparatus that use a browser software entity for specifying and downloading web pages from an internet-connected server. It should be realized, however, that the teachings of this invention apply as well to accessing and retrieving other data, such as database information, documents, and files, that is available through a network-connected server.

A method and system is disclosed for scheduling data download, such as web pages, databases or softwares, over a network such as the internet without keeping the computer system power on all the time till the upcoming data download activities. In FIG. 1 it shows when and how to set up the Internet Data Download Scheduler. It describes the following method from steps (a) to (d). In FIG. 2 it shows a method to be executed by the Internet Data Download Scheduler when the system timer is up at the scheduled wake-up time. It descriv=bes the following method from steps (e) to (k).

The method includes the steps of (a) initiating the data download request and the user input interfaces; saving the requesting computer system's network address, the requesting computer system being either connected to or disconnected from the network; (b) fetching from the internet (the network connected requesting computer system) or from the user's input (the network disconnected requesting computer system) and saving those web pages, databases or softwares' source entities and their corresponding network addresses (e.g. the web site URL on the distant server) for the upcoming data download (which web pages and their linked sub level pages are to be downloaded); (c) fetching and saving from the user's input on the data download schedules (when the data download is to take place) and user id and password if required for those web pages, databases or softwares for the upcoming download; and other user's options and choices, such as where the downloaded data is to be stored (i.e. cache or disk directory), limits on downloading time and allocated storage space, what the minimum bandwidth is between the requesting computer system and the distant server, and automatically rescheduling another data download for previous unsuccessful data download; data downloading either scheduled all at the same time in sequent or individually at different time; (d) setting the system timer at the wake-up time according to the data download schedules; (e) automatically turning on the requesting computer system according to the system timer if the requesting computer system is not turned on, and dialing up to connect to the network; generating a network busy signal to the Internal Data Download Schedule for rescheduling the data download according to the user's input of options and choices if the network is busy; (f) accessing the download data's network address and transmitting the data to the requesting computer system; providing the user id and password if required; (g) receiving and storing the requested web pages, databases or softwares in the requesting computer system for subsequent use by the user; transmitting successful and unsuccessful download messages from the destination entity to the requesting computer system; accordingly generating successful or unsuccessful data download messages for the user and unsuccessful data download signals to the Internet Data Download Scheduler to schedule another try of data download according to the user's input of options and choices; the user can access all these messages when the user logs on the requesting computer system; (h) interrupting a particular data downloading (transmitting and receiving) if the downloading time (estimated or actual) exceeding the user's earlier input downloading time limit or if the allocated download storage space reached the user specified limit; (i) automatically disconnecting from the network; (j) automatically rescheduling the data download in some other time, such as every hour afterwards, according to the user's earlier input choice if the previous data download is not successful or the network is too busy; (k) automatically turning off the requesting computer system. In this manner the requester doesn't need to keep the requesting computer system power on for the upcoming download actvities.

The most important advantage of this invention is that the requesting computer system does not have to be power on all the time till the upcoming download activities. Other advantages of this invention include the abilities of allowing the user to schedule data download from those web sites requiring an user id and password, and to specifying the upper limits on the download time and the allocated storage space for downloaded data.

In one embodiment the step of initiating includes steps of generating a web page download command and transmitting the web page download command to the destination entity. This may be accomplished over the network, or over another network, such as an intranet, that connects the initiating and destination entities. In this case the step of fulfilling includes initial steps of formulating, in response to receiving the web page download command at the destination entity, a network web page request message and transmitting the network web page request message from the destination entity to the web page source entity. A confirmation message may be sent to the initiating entity to confirm the receipt of the web page download command.

In another embodiment of this invention the step of initiating includes steps of generating the network web page request message that includes the user id and password if required and transmitting the network web page request message from the initiating entity to the web page source entity. In this case the web page source entity checks the user id and password if required before transmits the requested web page(s) to the destination entity at the requesting computer system network address.

The step of initiating includes a preliminary step of responding to a signal from a user through a user interface, such as by redefining mouse clicks when interacting with a web browser, such that the signal indicates that a specified web page is to be downloaded to and stored in the destination entity, as opposed to being fetched and displayed in a conventional manner. In this embodiment the step of responding includes a step of prompting the user to enter information for specifying at least one parameter related to downloading the web page, and/or includes a step of retrieving at least one user default parameter related to downloading the web page.

In a preferred embodiment the web page download command sent by the initiating entity includes a plurality of fields, including fields intended to specify: at least one user download preference; and at least one postprocessing operation to be performed on a received web page. The at least one user download preference includes at least one of: a number of web page levels to download; whether to download graphical data; a number of permissible retries to download a web page; and an interval between the retries. The at least one postprocessing operation includes at least one of: whether to decompress a received web page; whether to virus scan a received web page; and whether to print a received web page.

The step of receiving and storing the requested web page in the destination entity includes a step of writing data into an index entry associated with the received web page. The index entry is comprised of a plurality of fields, including fields intended to specify: the first and second network addresses, and a link summary of the web page. The index entry fields further specify at least one of: a time that the web page was downloaded; a number of bytes that were downloaded; a time that the web page download command was received by the destination entity; a number of retries that were required, if any, to download the web page; and an error report.

In a preferred embodiment of this invention the method includes a capability to transmit a cancellation message from the initiating machine to the destination machine. In response to receiving a cancellation message the destination machine one of terminates an on-going web page download, or deletes an already downloaded and stored web page, as well as the index information associated with the stored web page.

In a preferred, but not limiting embodiment, the network includes the internet, and the web page source entity is a WWW server compliant with conventional and/or extended HTTP protocols.

What is claimed is:

1. A method for downloading data from a server via a network, comprising the following steps:

initiating a data download request with a requesting entity having a first network address, said data comprising a web page, a database and a software;

said initiating said download request including the steps of:

generating a web page download command;

having an interface to allow said requesting entity to connect to said network;

responding to a signal from a user through a user interface as a preliminary step, said signal indicating that a specified web page is to be downloaded to and stored in said requesting entity;

initiating a user input interface including the steps of:

specifying a data source;

specifying a data network address;

specifying limits on downloading time;

specifying a limit on allocated storage space for said download data request;

specifying the minimum bandwidth between said requesting entity and a data server entity;

optionally specifying a user id and password;

fulfilling said data download request with a data server entity having a second network address;

transmitting said requested data to said requesting entity having said first network address;

receiving and storing said requested data in said requesting entity for subsequent access by a user;

there being a schedule for said receiving and storing of said requested data in said requesting entity for subsequent access by a user which is selected from the group consisting of:

data downloading scheduled all at the same time in sequence;

data downloading scheduled individually at different times;

the step of initiating a user input interface also including the steps of:

specifying cache or disk directory where the downloaded data is to be stored;

automatically rescheduling another data download for previous unsuccessful data download; and the fulfilling said data download request includes transmitting the network web page request message from said initiating entity to the web page server entity; and

ending the down load request.

2. A method as in claim **1**, wherein the data is comprised of:

a web page;

a database; and

a software.

3. A method as in claim **1**, wherein the step of initiating a data download request includes steps of:

generating a web page download command; the requesting entity having an interface for connecting to the network;

a preliminary step of responding to a signal from a user through a user interface, the signal indicating that a specified web page is to be downloaded to and stored in the requesting entity.

4. A method as in claim 1, wherein the step of intiating a user input interface includes steps of:

specifying a data source;

specifying a data network address; and

specifying a user id and password if required;

specifying limits on downloading time;

specifying a limit on allocated storage space for said download data request;

specifying the minimum bandwidth between the requesting entity and the data server entity and

specifying the data download schedule.

5. A method as in claim 4, wherein the data download schedule is selected from the group consisting of:

data downloading scheduled all at the same time in sequent; and

data downloading scheduled individually at different times.

6. A method as in claim 4, further including specifying the other user's options and choices selected from the group consisting of one or more of:

specifying cache or disk directory where the downloaded data is to be stored; and

automatically rescheduling another data download for previous unsuccessful data download;

supplying the user id and password if required; and

where the step of fulfilling includes initial steps of,

transmitting the network web page request message from the initiating entity to the web page server entity.

7. A method as in claim 1, wherein the data download command is transmitted over the network or a second network separate from the network.

8. A method as in claim 1, wherein the step of responding includes a step of prompting the user to enter information for specifying at least one parameter related to downloading the data.

9. A method as in claim 1, wherein the step of responding includes a step of retrieving at least one user default parameter related to downloading the data.

10. A method as in claim 1, wherein the data download command is comprised of a plurality of fields, including fields for specifying:

at least one user download preference.

11. A method as in claim 10, wherein the at least one user download preference includes at least one of:

nummber of web page levels to download;

whether to download graphical data;

a number of retries to download a web page; and

an interval between retries.

12. A method as in claim 10, wherein the at least one postprocessing operation includes at least one of:

whether to decompress a received web page;

whether to virus scan a received web page; and

whether to print a received web page.

13. A method as in claim 1, wherein the step of fulfilling the data download request, comprising the steps of:

setting the system timer at the wake-up time according to the data download schedules;

automatically turning on the requesting computer system according to the system timer if the requesting computer system is not turned on; and

dialing up to connect to the network;

generating a network busy signal to the Internet Data Download Schedule for rescheduling the data download according to the user's input of options and choices if the network is busy.

14. A method as in claim 1, wherein the step of transmitting the requested data to the requesting entity, comprising the steps of:

providing the user id and password if required;

transmitting successful and unsuccessful download messages from the destination entity to the requesting computer system;

accordingly generating successful or unsuccesful data download messages for the user and unsuccessful data download signals to the Internet Data Download Scheduler to schedule another try of data download according to the user's input of options and choices;

interrupting a particular data download if the downloading time exceeds the user's earler input downloading time limit or if the allocated download storage space reached the user specified limit;

interrupting a particular data download if the banwidth between the requesting computer system and the distant server drops below the user specified minimum bandwidth.

15. A method as in claim 1, wherein the step of receiving and storing the requested web page in the destination entity includes a step of writing data into an index entry associated with the received web page.

16. A method as in claim 15, wherein the index entry is comprised of a plurality of fields, including fields for specifying:

the first and second network addresses; and

a link summary of the web page.

17. A method as in claim 16, wherein the fields further specify at least one of:

a time that the web page was downloaded;

a number of bytes that were downloaded;

a time that the web page download command was received by the destination entity;

a number of retries that were required, if any, to download the web page; and

an error report.

18. A method as in claim 1, and further comprising steps of:

transmitting a cancellation message from the initiating entity to the destination entity;

receiving the cancellation message; and

in response, one of terminating an on-going data page download or deleting an already downloaded and stored data.

19. A method as in claim 1, wherein the ending of the data download, comprising the steps of:

automatically disconnecting from the network;

automatically rescheduling the data download in some other time, such as every hour afterwards, accoding to the user's earlier input choice if the previous data download is not successful or the network is too busy; and

automatically turning off the requesting computer system.

20. A method as in claim 1, wherein the network includes the internet, and wherein the server entity is comprised of a WWW server.

21. A method as in claim 14, wherein the particular data download is selected from the group consisting of transmitting and receiving.

22. A method as in claim 18, wherein a user can access all these messages when the user logs on the requesting computer system.

* * * * *